COMP GI22/MI22
Deep Learning Lecture 1

Thore Graepel & Guest Lecturers from DeepMind
t.graepel@ucl.ac.uk

加入知识星球资源管理库，每日免费获取报告

1、每天分享30+最新行业报告（涵盖科技、金融、教育、互联网、房地产、生物制药、医疗健康等最新行业）

2、每天分享10+企业咨询管理文件（涵盖国内外著名咨询公司相关管理方案，企业运营制度等）

3、每天分享10+科技类论文或者大师课件、笔记。

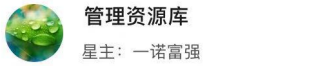
4、每天更新企业运营管理中遇到的问题及解决方案

5、高端人才管理与行业交流

更多文件请点击链接， https://t.zsxq.com/2nElq7a

微信扫码进微信群，每日获取资料（微信号：Teamkon）

微信二维码



管理资源库

Overview

- Team and Structure of the Course
- Guest Lectures and Lecturers
- DeepMind approach to AI
- Why Deep Learning?
- Deep Reinforcement Learning at work
  - Learning to Play Atari Games with Deep RL
  - AlphaGo - Learning to master level Go
- Extra revision material (supervised learning)

The DeepMind/UCL Team

Koray Kavukcuoglu
(Co-Lead DL)

Hado van Hasselt
(Co-Lead RL)

Matteo Hessel

Diana Borsa
(TA Lead)

Teaching Assistants:

- Zach Eaton-Rosen
- Lewis Moffat
- Michael Jones
- Raza Habib
- Thomas Gaudelet

Alex Davies

Marie Mulville
(PgM)

Format and Coursework

- Format: Two streams, both streams mandatory
  - **Tuesdays:** Deep Learning taught by a selection of fantastic guest lecturers from DeepMind
  - **Thursdays:** Reinforcement Learning taught by Hado Van Hasselt (also DeepMind)
  - Some exceptions, check timetable at https://timetable.ucl.ac.uk/ and on Moodle (for topics)
- Assessment: 100% through Coursework
  - There are **four deep learning** and **four reinforcement learning assignments**
  - Each of the **eight assignment will be weighted equally**, i.e., counting 12.5%
  - Coursework is **mixture of programming assignments and questions**
  - Framework for coursework will be **Colab**, a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.
  - Machine Learning algorithms will be implemented in **TensorFlow** through Colab.
  - You can find more **information about the assessment on Moodle.**
  - **Todo:** Set up Google account with address: "ucl.compgi22.2018.XXXXXXXX@gmail.com", where XXXXXXXX is your (numerical) student number
- Support: Use Moodle forum and Moodle direct messages

Welcome to Colaboratory!

Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

Colaboratory notebooks are stored in Google Drive and can be shared just as you would with Google Docs or Sheets. Colaboratory is free to use.

For more information, see our FAQ.

Python 3 (NEW!)

Colab now supports both Python2 and Python3 for code execution.

- When creating a new notebook, you'll have the choice between Python 2 and Python 3.
- You can also change the language associated with a notebook; this information will be written into the `.ipynb` file itself, and thus will be preserved for future sessions.

```
[ ]  import sys
     print('Hello, Colaboratory from Python {}!'.format(sys.version_info[0]))
```

```
Hello, Colaboratory from Python 3!
```

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

[1. 1. 1.] [1. 2. 3.] [2. 3. 4.]

http://tensorflow.org/

and

https://github.com/tensorflow/tensorflow

Open, standard software for general machine learning

Great for Deep Learning in particular

First released Nov 2015

Apache 2.0 license

Warning: Lots of work and prior knowledge required!

- Last year, many people complained that it was too much work!
- If you do not know how to code in Python this may not be right for you!
- A lot of preliminary knowledge required - see quiz!
- Deep Learning lectures are delivered by top researchers in the field and will stretch towards the current research frontier → brace yourselves!
- Check out the Self-Assessment Quiz on Moodle

		Advanced Deep Learning and Reinforcement Learning (COMPGI22)						
		Deep Learning Path Outline			Reinforcement Learning Path Outline		Coursework	
Date Tuesday 2-4pm	Week	Title	Lecturer	Date Thursday 9-11pm	Topic	Lecturer	OUT (Start of the week)	DUE (End of week)
9/1/18	1	DL1: Introduction to Machine Learning based AI	Thore Graepel	11/1/18	DL2: Introduction to Tensorflow	Matteo Hessel & Alex Davies		
16/1/18	2	DL3: Neural Networks Foundations	Simon Osindero	18/1/18	RL1: Intro to reinforcement learning	Hado Van Hasselt	DLCW1	
23/1/18	3	DL4: Convolutional Neural Networks for Image Recognition	Karen Simonyan	25/1/18	RL2: Bandits and exploration	Hado Van Hasselt	DLCW2	DLCW1
30/1/18	4	RL3: Model-Free Prediction	Hado Van Hasselt	01/02/18	RL4: MDPs and Dynamic Programming	Hado Van Hasselt	RLCW1	DLCW2
06/02/18	5	DL5: Sequences and Recurrent Networks	Oriol Vinyals	08/02/18	DL6: Beyond image recognition, end-to-end learning, embeddings	Raia Hadsell	RLCW2	
	Reading Week				Reading Week		DLCW3	RLCW1
20/02/18	6	RL5: Model-Free Control	Hado Van Hasselt	22/02/18	RL6: Policy Gradient Methods	Hado Van Hasselt		RLCW2
27/02/18	7	DL7: Optimization for Machine Learning	James Martens	29/02/18	RL7: Planning and models	Hado Van Hasselt	RLCW3	DLCW3
06/03/18	8	DL8: Deep Learning for Natural Language Processing	Ed Grefenstette	08/03/18	RL8: Building full agents	Hado Van Hasselt	DLCW4	RLCW3
13/03/18	9	DL9: Attention and Memory in Deep Learning	Alex Graves	15/03/18	RL9: Guest talk	Vlad Mnih (TBC)	RLCW4	
20/03/18	10	DL10: Unsupervised Lerning and Generative Models	Shakir Mohamed	22/03/18	RL10: Guest talk	David Silver (TBC)		DLCW4
								RLCW4

DeepMind Guest Lecturers

Introduction to TensorFlow

- Lecture topics:
  - Introduction to Tensorflow principles
  - Practical work-through examples in Colab
- Guest Lecturer: Matteo Hessel
  - Joined DeepMind in 2015.
  - Masters in Machine Learning from UCL
  - Master of Engineering Politecnico di Milano
- Guest Lecturer: Alex Davies
  - Joined Deepmind in 2017
  - PhD in Machine Learning at Cambridge
  - Worked with team of international scientists to build the world's first machine learned musical.

Matteo Hessel

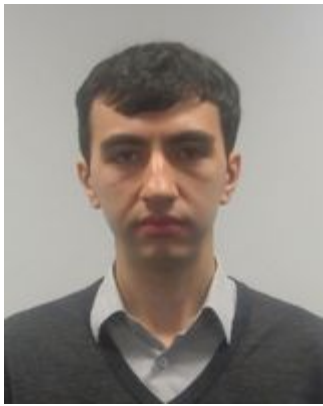
Alex Davies

Neural Nets, Backprop, Automatic Differentiation

- Lecture topics:
  - Neural nets
  - Multi-class classification and softmax loss
  - Modular backprop
  - Automatic differentiation
- Guest Lecturer: **Simon Osindero**
  - Joined DeepMind in 2016.
  - Undergrad/Masters in Natural Sciences/Physics at University of Cambridge.
  - PhD in Computational Neuroscience from UCL (2004). Supervisor: Peter Dayan.
  - Postdoc at University of Toronto with Geoff Hinton. (Deep belief nets, 2006).
  - Started an A.I. company, LookFlow, in 2009. Sold to Yahoo in 2013.
  - Current research topics: deep learning, RL agent architectures and algorithms, memory, continual learning.

Convolutional Neural Networks

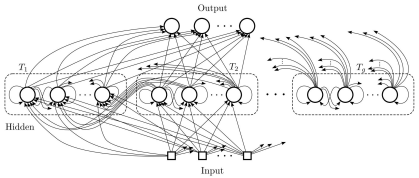
- Lecture topics:
  - Convolutional networks
  - Large-scale image recognition
  - ImageNet models
- Guest Lecturer: **Karen Simonyan**
  - Joined DeepMind in 2014
  - DPhil (2013) and Postdoc (2014) at the University of Oxford with Andrew Zisserman
  - Research topics: deep learning, computer vision
    - VGGNets, two-stream ConvNets, ConvNet visualisation, etc.
    - https://scholar.google.co.uk/citations?user=L7lMQkQAAAAJ

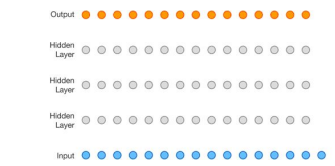


Recurrent Nets and Sequence Generation

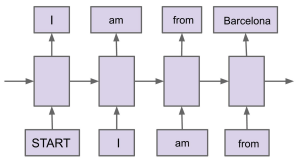
- Lecture topics:
  - Recurrent Neural Networks
  - Long-Short Term Memory (LSTM)
  - (Conditional) Sequence Generation
- Guest Lecturer: **Oriol Vinyals**
  - Joined DeepMind in 2016.
  - Worked in Google Brain from 2013 to 2016.
  - PhD in Artificial Intelligence from UC Berkeley (2009-13). Supervisor: Darrell / Morgan.
  - Current research topics: deep learning, sequence modeling, generative models, distillation, RL/Starcraft, one shot learning.

Temporal Hierarchies

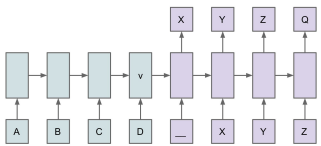




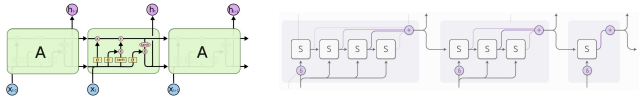
Sequence Prediction



Seq2Seq



Recurrent Architectures



End-To-End and Energy-Based Learning

- Lecture topics:
  - End-to-end learning
  - Energy based learning
  - Ranking
  - Embeddings
  - Triplet loss
- Guest Lecturer: **Raia Hadsell**
  - PhD From NYU, postdoc at CMU's Robotics Institute
  - Senior Scientist and Tech Manager at SRI International
  - Now leading a research team at DeepMind
  - Research in Deep Learning, Robotics, Navigation, Life-Long Learning

Optimisation

- Lecture topics:
  - First-order methods
  - Second-order methods
  - Stochastic methods
  - Some convergence theory
- Guest Lecturer: **James Martens**
  - Joined DeepMind in Sept 2016
  - PhD from University of Toronto under Geoff Hinton & Rich Zemel in 2015
  - Undergrad from Waterloo in Math and Computer Science
  - Working on: second-order optimization for neural nets, characterizing expressive power/efficiency of neural nets, generative models / unsupervised learning

Attention and Memory Models

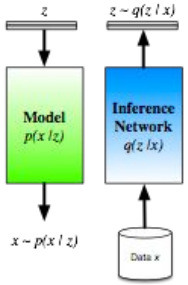
- Lecture topics:
  - Neural attention models
  - Recurrent neural networks with external memory
  - Neural Turing Machines / Differentiable Neural Computers
- Guest Lecturer: **Alex Graves**
  - Joined Deepmind 2013
  - Undergrad Theoretical Physics, Univ. of Edinburgh
  - Masters Mathematics and Theoretical Physics, Univ. of Cambridge
  - PhD Artificial Intelligence TU Munich, supervisor Jürgen Schmidhuber
  - CIFAR Junior fellow with Geoff Hinton, Univ. of Toronto
  - Research focuses on sequence learning with recurrent neural networks: memory, attention, sequence generation, model compression

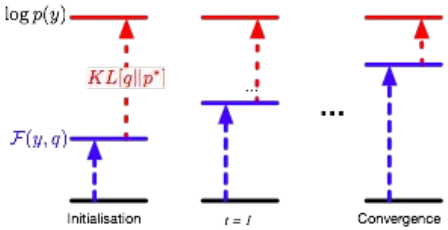
Deep Learning for Natural Language Processing

- Lecture topics:
  - Deep Learning for Natural Language Processing
  - Neural word embeddings
  - Neural machine translation
- Guest Lecturer: **Ed Grefenstette**
  - DPhil from Oxford
  - Co-Founder of Dark Blue Labs (acquired by DeepMind)
  - Research in Machine Learning, Computational Linguistics

Unsupervised Learning and Deep Generative Models

- Lecture topics:
  - Density estimation and unsupervised learning.
  - Deep Generative Models: latent variable and implicit models.
  - Approximate inference and variational inference.
  - Stochastic optimisation
- Guest Lecturer: **Shakir Mohamed**
  - Joined DeepMind in 2013.
  - PhD in Statistical Machine Learning, St John's College, University of Cambridge. Supervisor: Zoubin Ghahramani.
  - CIFAR Junior Research Fellow at the University of British Columbia with Nando de Freitas.
  - Topics in Probabilistic thinking, approximate Bayesian inference, unsupervised learning and density estimation, deep Learning, reinforcement learning.
  - Undergrad in electrical engineering. From Johannesburg, South Africa.





Integral problem $$p(y) = \int p(y|z)p(z)dz$$

Reinforcement Learning Stream (Hado)

- Introduction to Reinforcement Learning
- Markov Decision Processes
- Planning by Dynamic Programming
- Model-Free Prediction
- Model-Free Control

- Value Function Approximation (Deep RL)
- Policy Gradient Methods
- Integrating Learning and Planning
- Exploration and Exploitation
- Case Study: AlphaGo

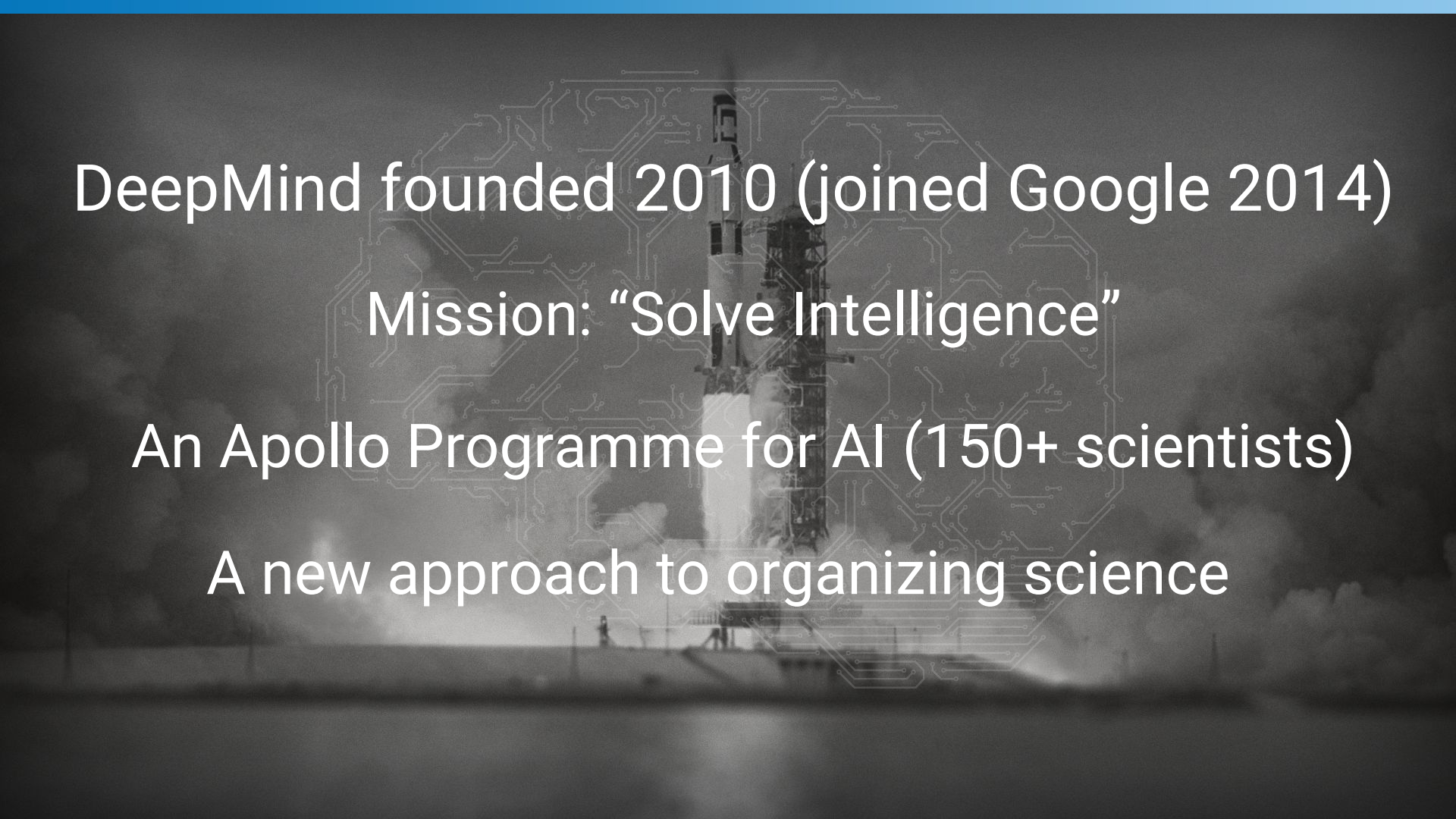
Hado van Hasselt

Case Study: AlphaGo (TBC)

- Lecture topics:
  - The story behind AlphaGo
  - Deep RL applied to Classical Board Games
  - Combining Tree Search and Neural Networks
  - Evaluation against machines and humans
- Guest Lecturer: **David Silver**
  - Computer Science at Cambridge, PhD Alberta
  - Co-Founder/CTO of Elixier Studios
  - Faculty member at UCL (on leave at DeepMind)
  - Joined DeepMind in 2013
  - Research in deep reinforcement learning, integration of learning and planning, games

Case Study: Practical Deep RL (TBC)

- Lecture topics:
  - Learning to play Atari games: DQN in Detail
  - Faster Agents through parallel training
  - Better data efficiency through unsupervised RL
  - Some practical advice
- Guest Lecturer: **Volodymyr Mnih**
  - PhD in Machine Learning at the University of Toronto
  - Early DeepMind pioneer
  - Legendary work on Deep RL for playing Atari, published in Nature


DeepMind founded 2010 (joined Google 2014)
Mission: “Solve Intelligence”
An Apollo Programme for AI (150+ scientists)
A new approach to organizing science

General-Purpose Learning Algorithms

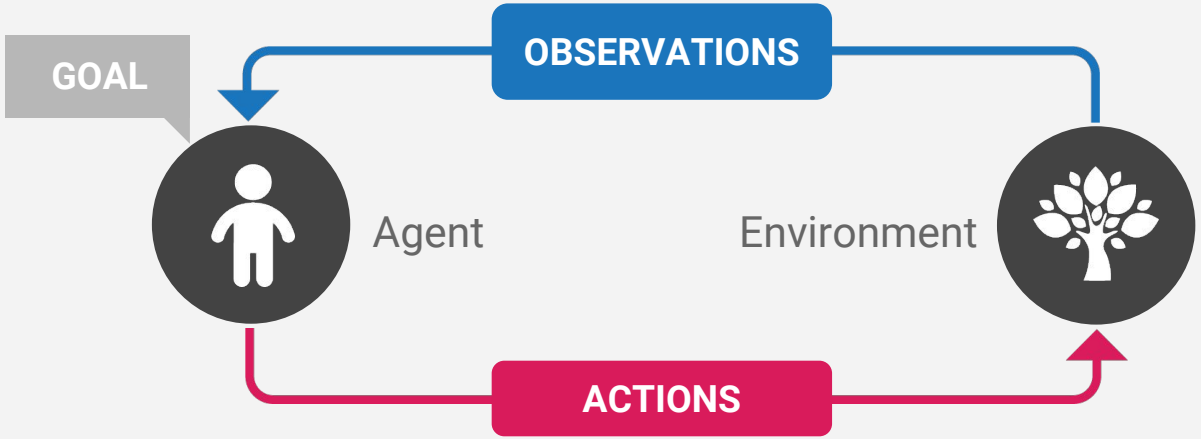
Learn automatically from raw inputs - not pre-programmed

General - same system can operate across a wide range of tasks

Artificial 'General' Intelligence (AGI) – flexible, adaptive, inventive

'Narrow' AI – hand-crafted, special-cased, brittle

Reinforcement Learning



- General Purpose Framework for AI
- Agent interacts with the environment
- Select actions to maximise long-term reward
- Encompasses supervised and unsupervised learning as special cases

What is intelligence?

Intelligence measures an agent's ability to achieve goals in a wide range of environments

$$\Upsilon(\pi) := \sum_{\mu \in E} 2^{-K(\mu)} V_\mu^\pi .$$

Measure of Intelligence

Sum over environments

Complexity penalty

Value achieved

Universal Intelligence: A Definition of Machine Intelligence, Legg & Hutter 2007

Grounded Cognition

A true thinking machine has to be grounded in a rich sensorimotor reality

Games are the perfect platform for developing and testing AI algorithms

Unlimited training data, no testing bias, parallel testing, measurable progress

‘End-to-end’ learning agents: from pixels to actions

Why Deep Learning?

Thanks to Koray for DL slides

- Enables End-To-End Training
  - Optimise for the end loss
  - Don't engineer your inputs
  - Learn good representations
- Versatile: Can be applied to images, text, audio, video
- Modular design of systems (modular backprop)
- Represent weak prior knowledge (e.g., convolutions)
- Now computationally feasible at scale (GPUs)

Supervised Learning

- Convolutional Networks on **MNIST**

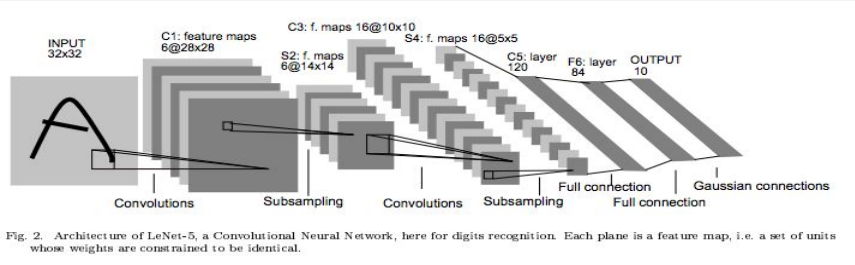
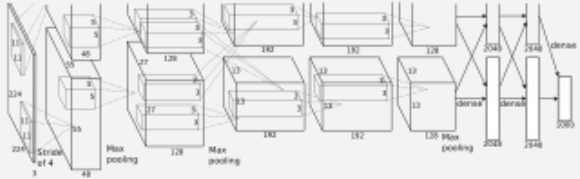
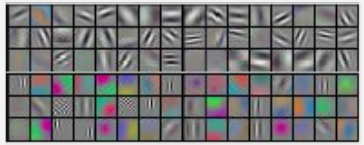


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

[Lecun, et. al]

- Convolutional Networks on **ImageNet**

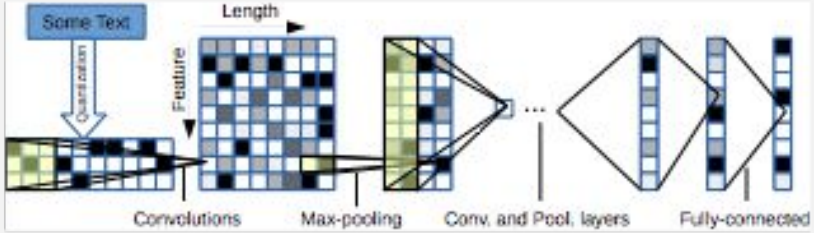




[Krizhevsky, et. al]

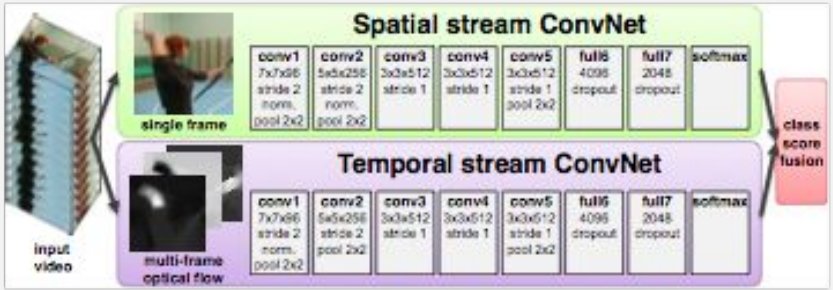
Supervised Learning

- Convolutional Networks on **Text**

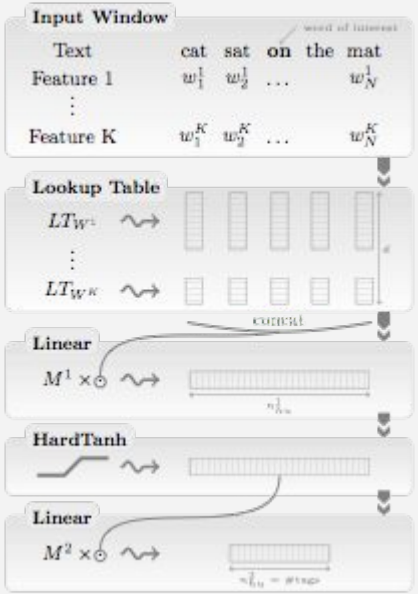


[Zhang, et. al]

- Convolutional Networks on **Video**



[Simonyan, et. al]



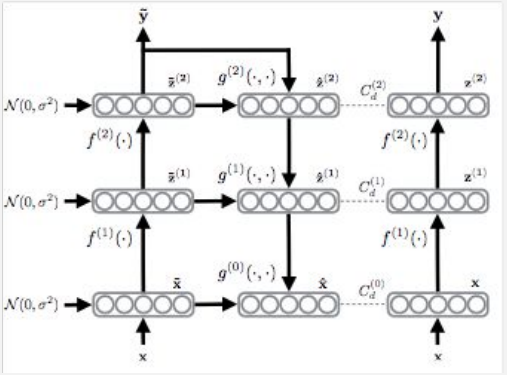
[Collobert, et. al]

Supervised Learning

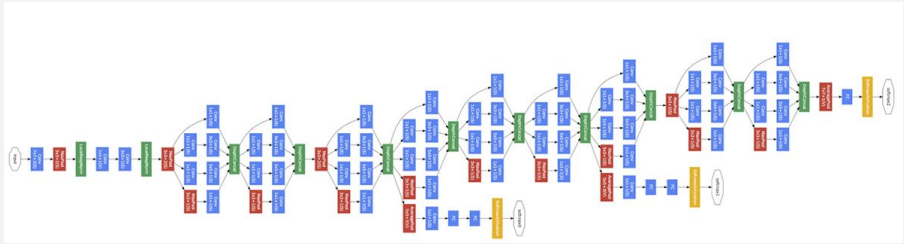
- End-to-End Training
- **Optimize** for the **end loss**
- **No engineered** inputs
- With enough data, **learn a big non-linear function**
- Learn **good representations** of data
  - Rich enough supervised labeling is enough to train transferrable representations
  - Best feature extractor
  - Karpathy, Razavian et al, Yosinski et al, Donahue et al
- Large labeled dataset + big/deep neural network + GPUs
- Ever more sophisticated modules → Differentiable Progrogramming

Supervised Learning

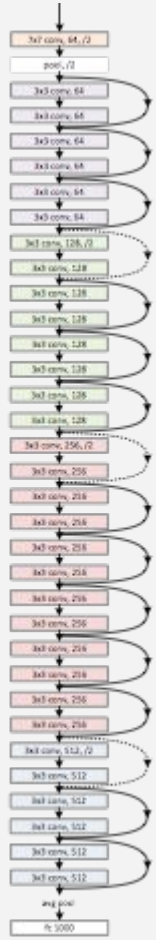
- Innovation continues
  - Inception
  - Ladder Nets
  - Residual Connections
  - …
- Performance is continuously improving
- Architectures for easier optimization
  - Batchnorm



[Rasmus, et. al]



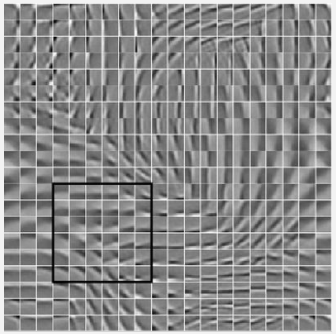
[Szegedy, et. al]



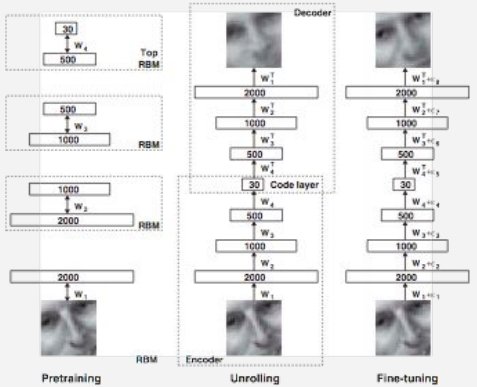
[He, et. al]

Unsupervised Learning

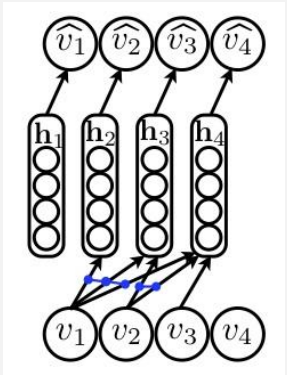
- Unsupervised Learning/Generative Models
  - RBM
  - Auto-encoders
  - PCA, ICA, Sparse Coding
  - VAE
  - NADE - and all variants
  - GANs



- How to evaluate/rank different algorithms?
- Quantitative approach or visual quality?
  - How can we trust if the input domain itself is not interpretable?
- How can unsupervised learning help a task?



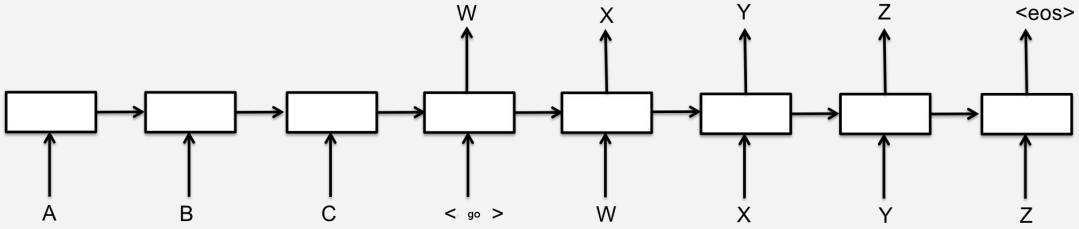
[Hinton, et. al]



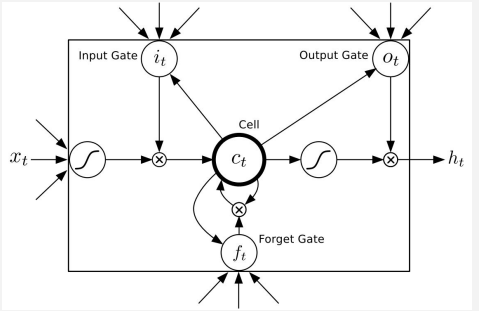
[Larochelle, Murray]

Sequence Modeling

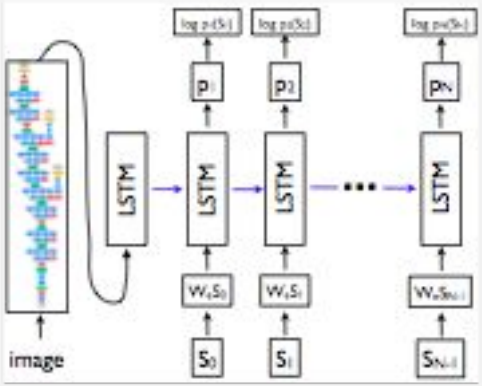
- Almost all data are sequence
  - Text
  - Video
  - Audio
  - Image [nade, pixelrnn]
  - Multi-modal (caption → image, image → caption)



[Sutskever, et. al]



[Hochreiter and Schmidhuber]



[Vinyals, et. al]

Human-level control through deep reinforcement learning

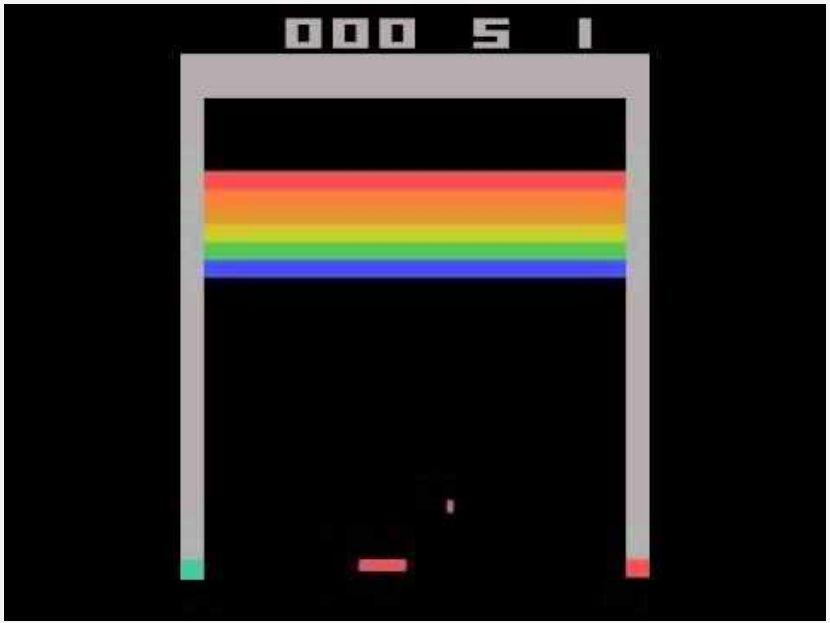
Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis

Google DeepMind

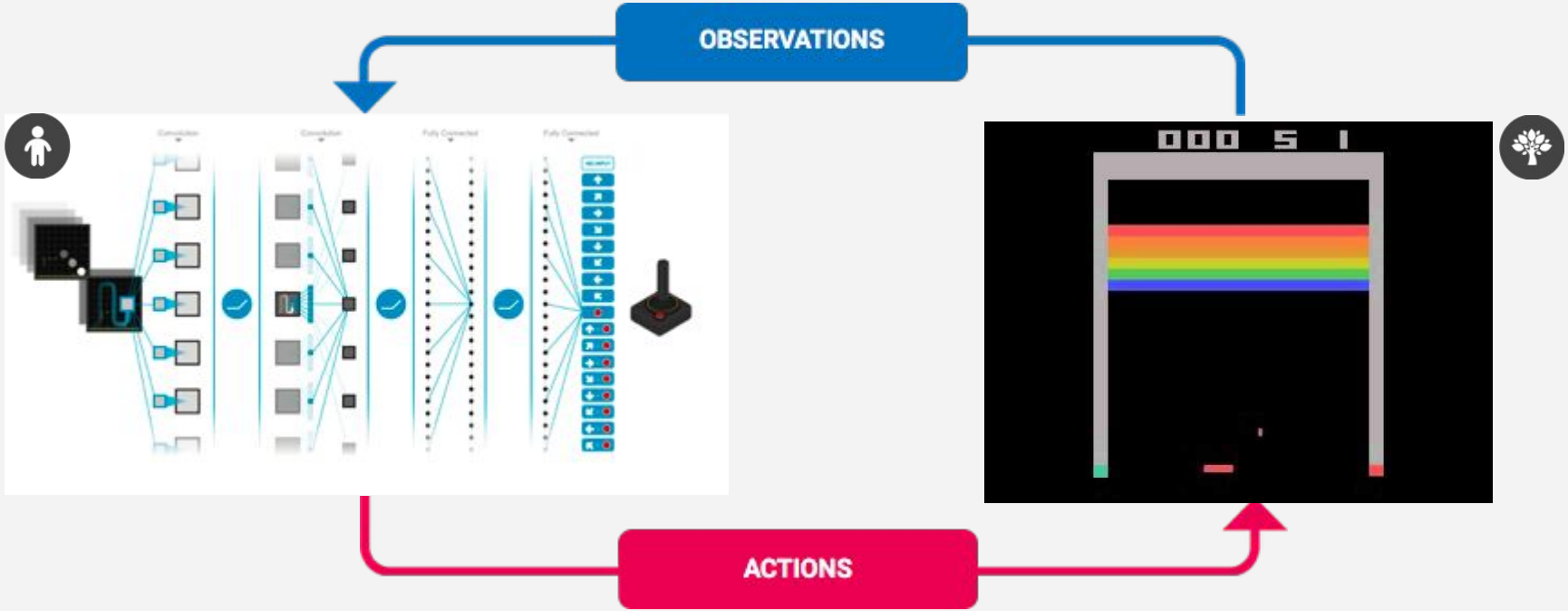
(Mnih et al. Nature 2015)

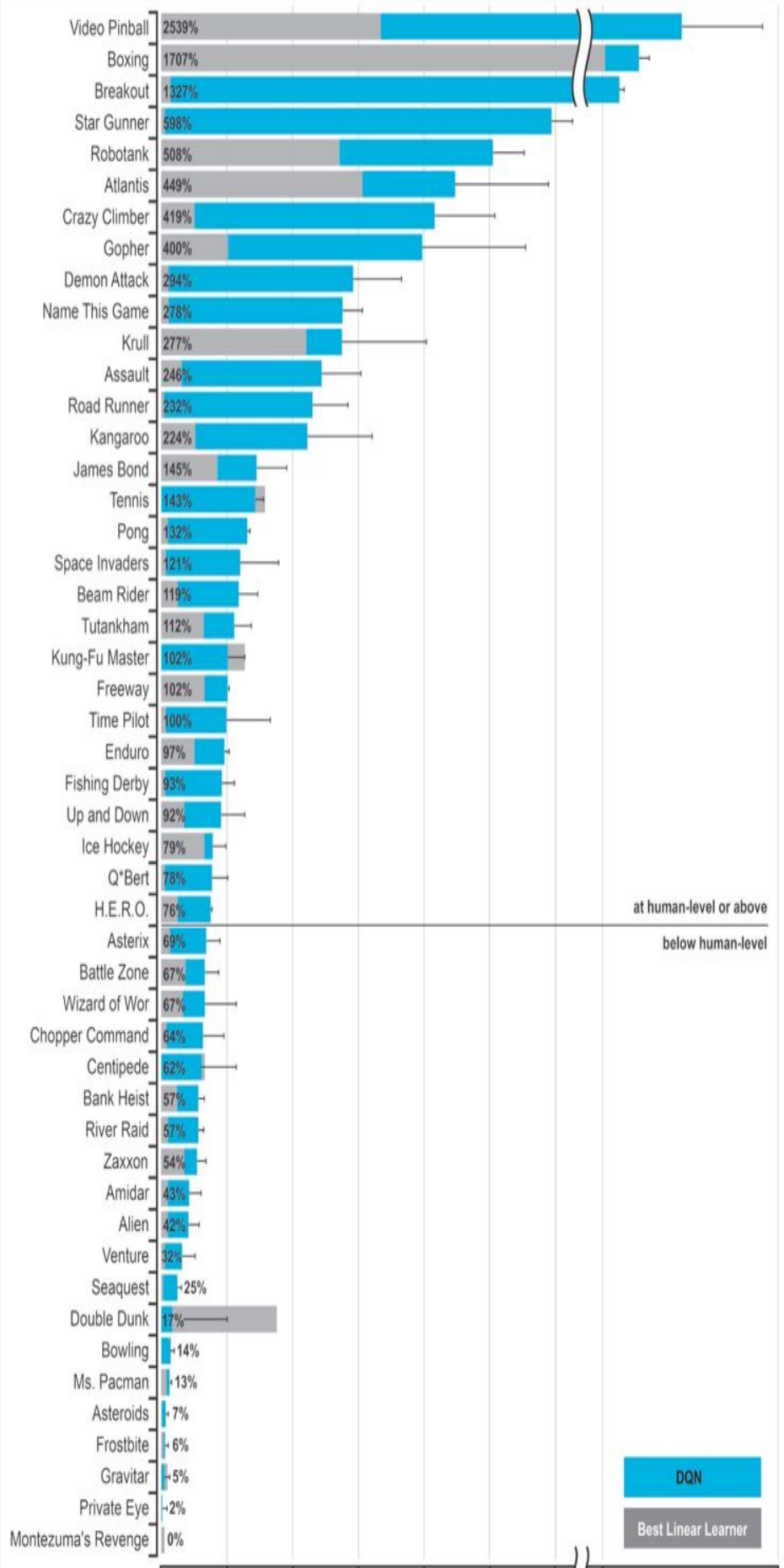
ATARI Games

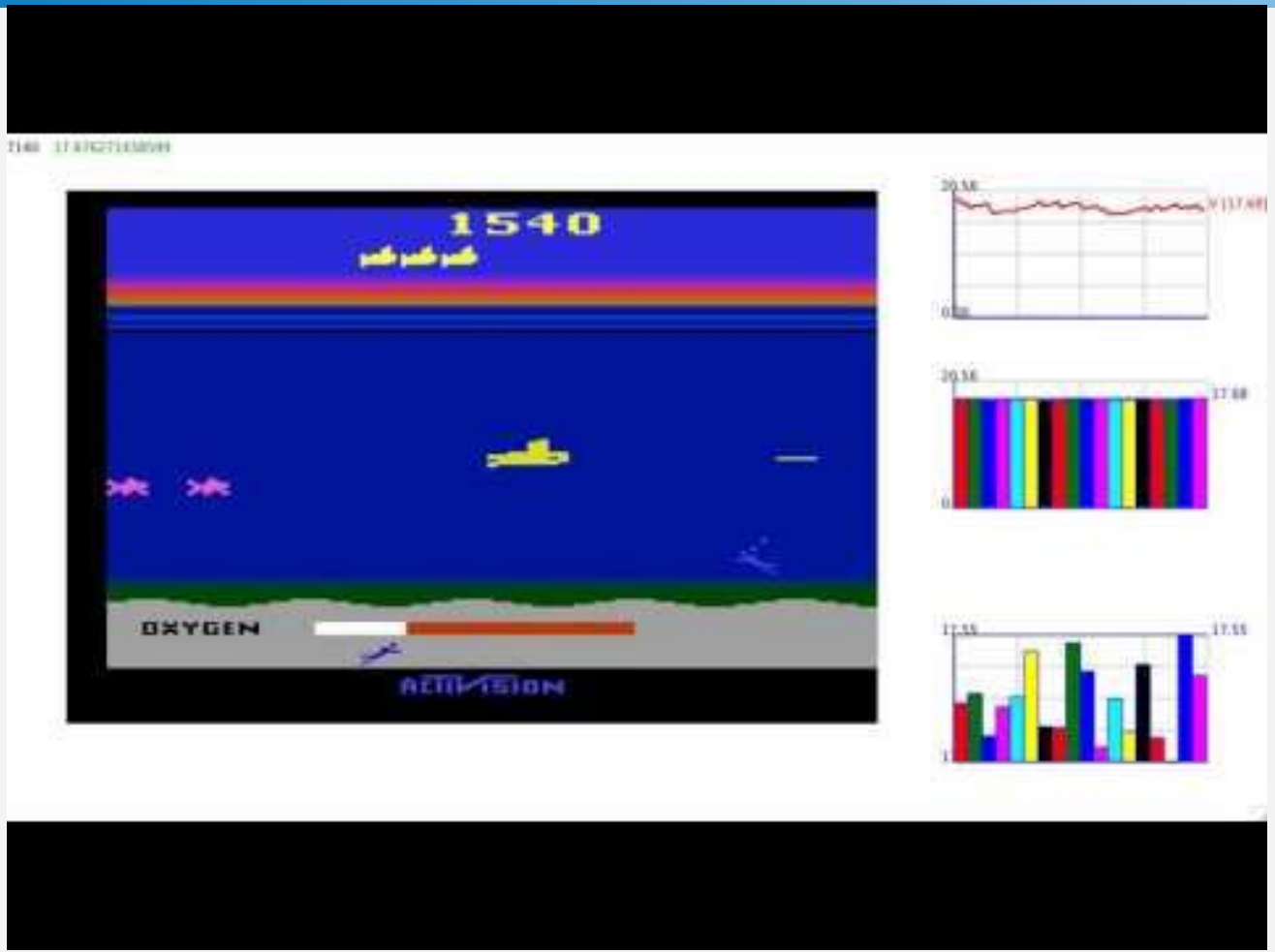
- Designed to be **challenging** and interesting for humans
- Provides a **good platform** for sequential decision making
- Widely adopted **RL benchmark** for evaluating agents (Bellemare'13)
- Many **different games** emphasize control, strategy, …
- Provide a rich visual domain

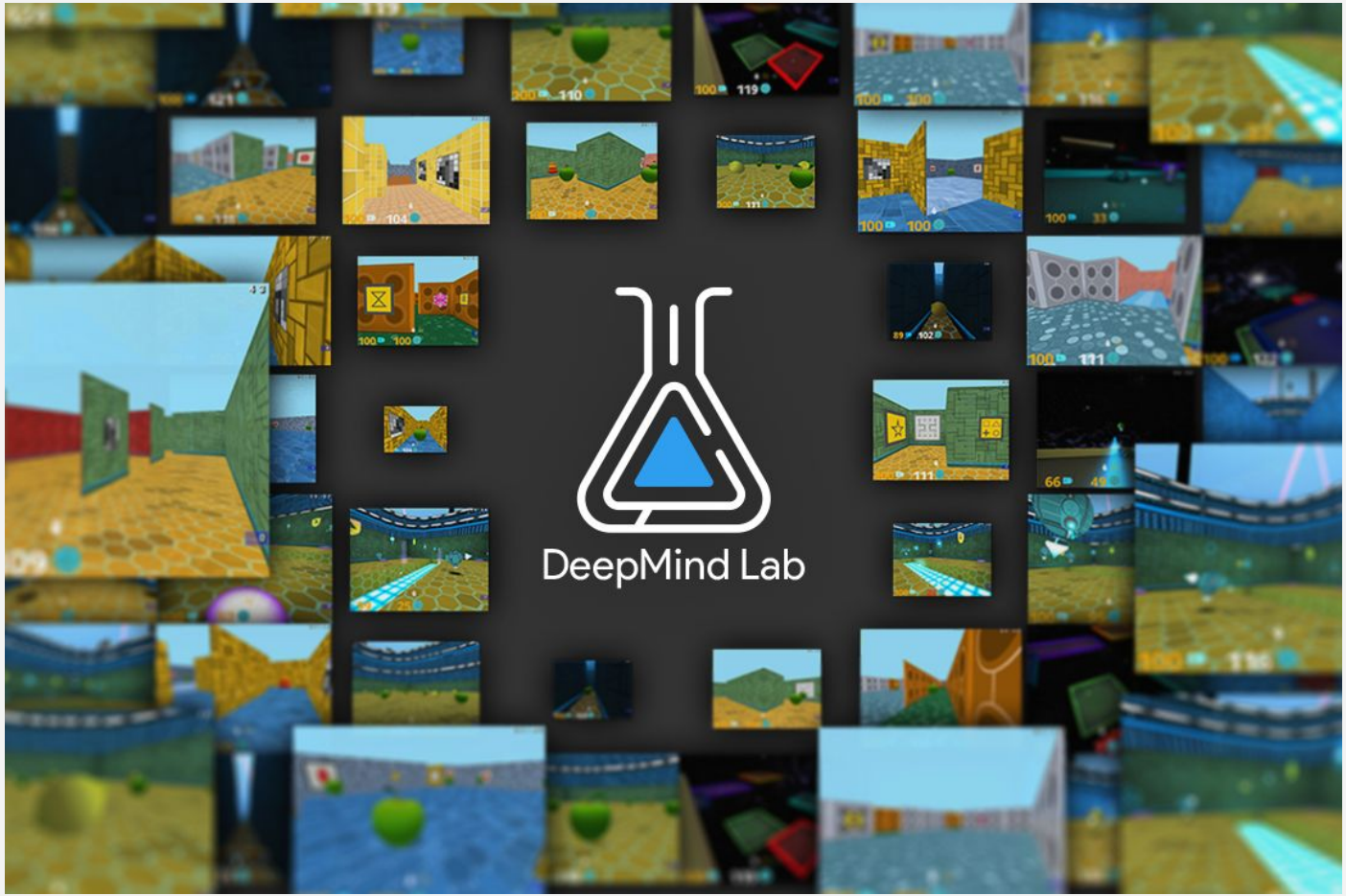


End-to-End Reinforcement Learning

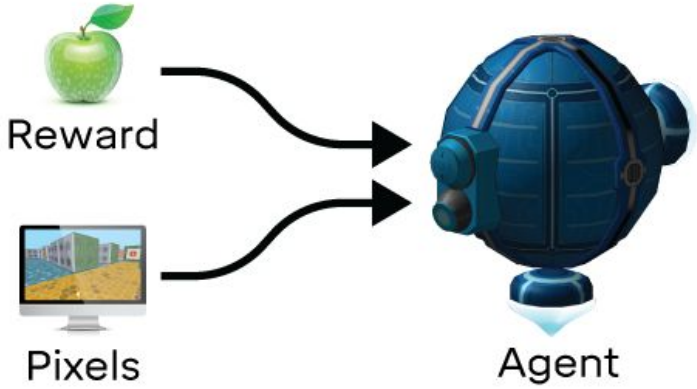



Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*Bert 78%
H.E.R.O. 76%
at human-level or above
Asterix 69%
below human-level
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pacman 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
DQN
Best Linear Learner
0%
100%
200%
300%
400%
500%
600%
1000%
4500%

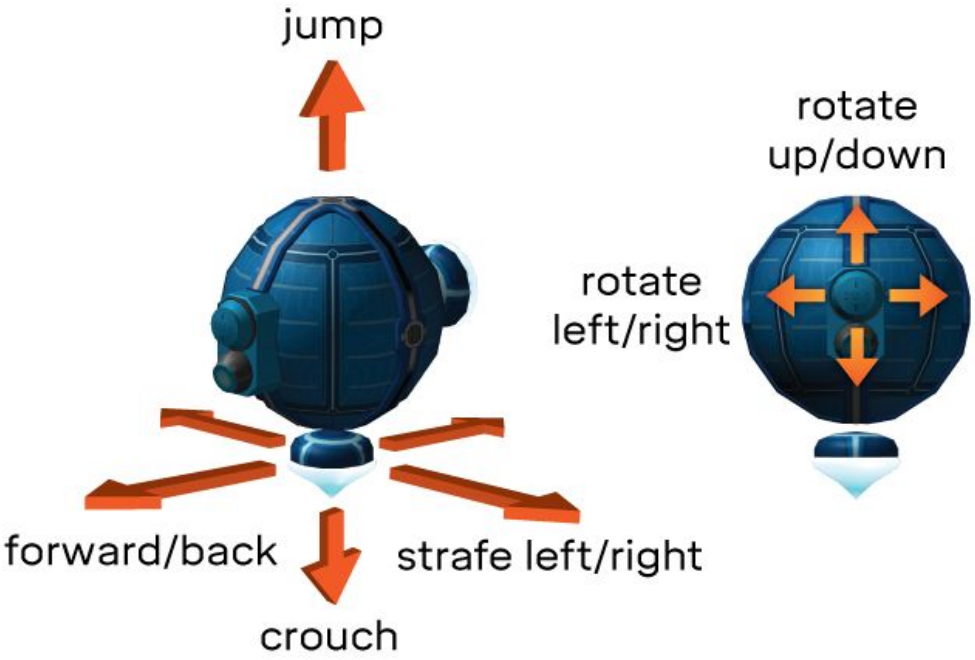

1540
OXYGEN
ACTIVISION


DeepMind Lab

Observations



Actions



DeepMind Lab - Challenging RL Problems in 3D

Mastering the game of Go with deep neural networks and tree search

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

Google DeepMind

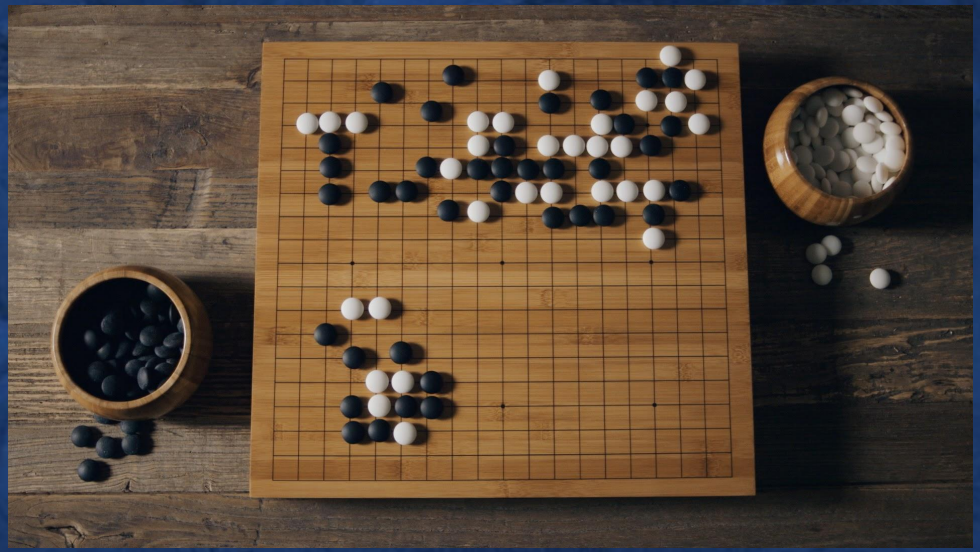
(Silver, Huang, et al 2016)
#3 most downloaded academic paper this month

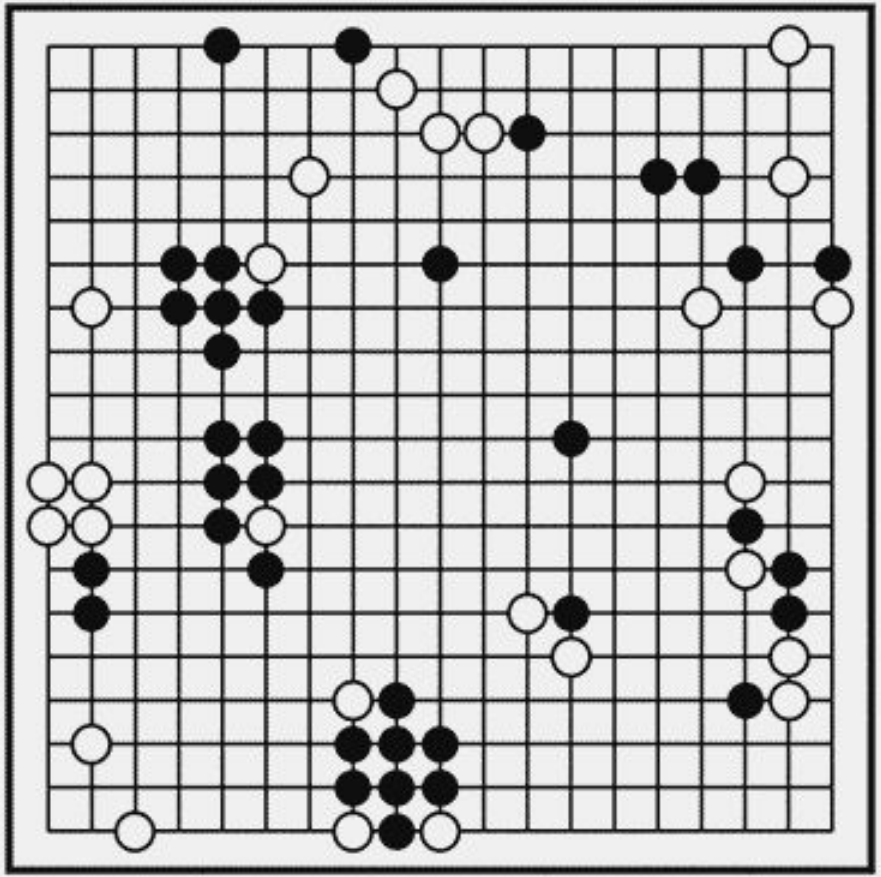
Why is Go hard for computers to play?

Game tree complexity = b^d

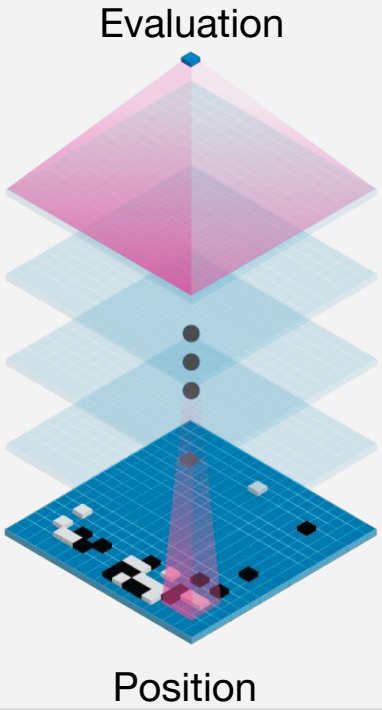
Brute force search intractable:

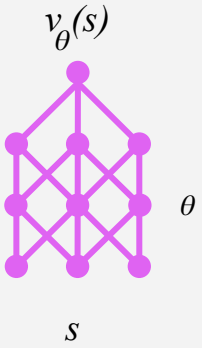
1. Search space is huge
2. “Impossible” for computers to evaluate who is winning





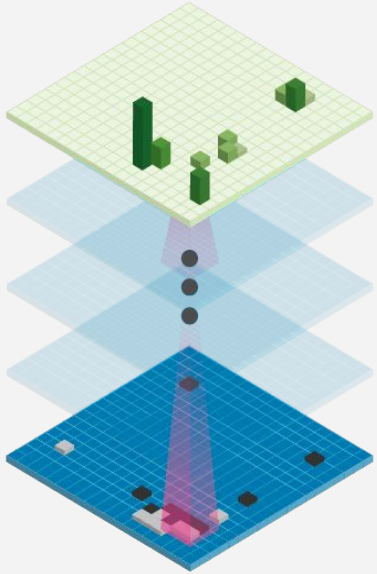
Value network





Policy network

Move probabilities



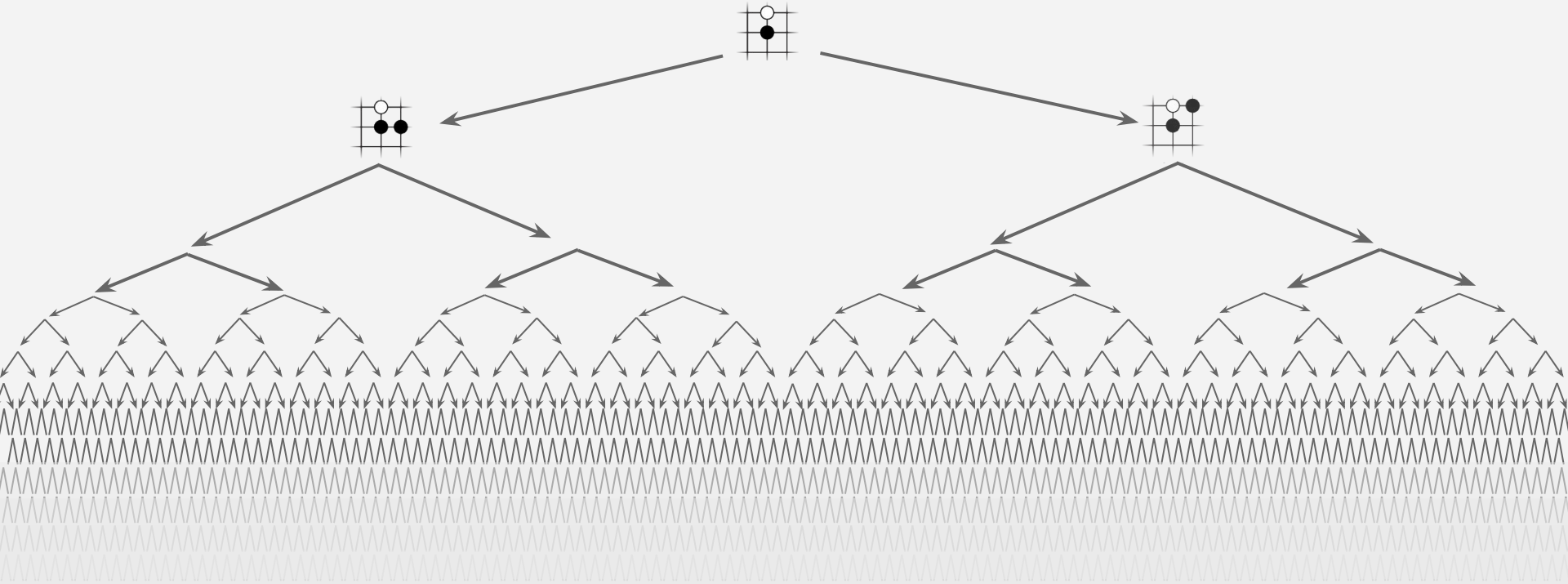
Position

$p_\sigma(a|s)$

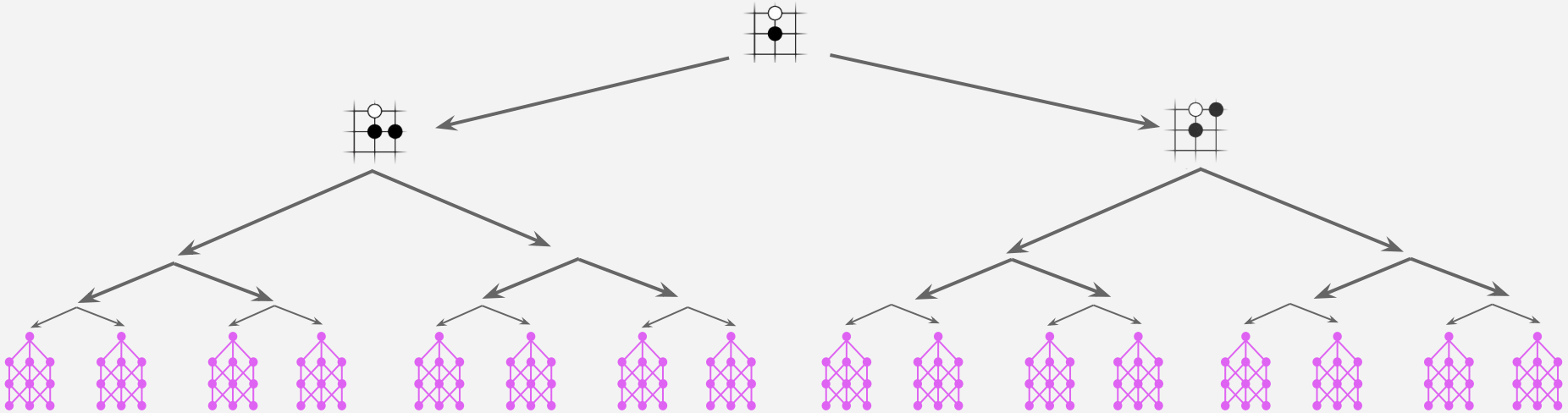
σ

s

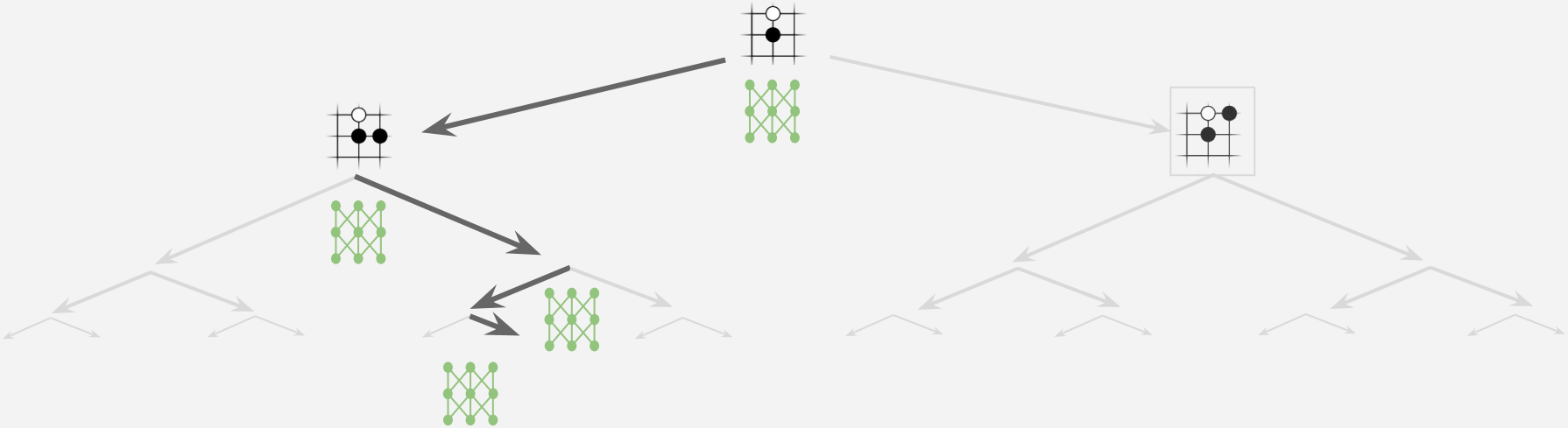
Reducing depth with value network



Reducing depth with value network



Reducing breadth with policy network

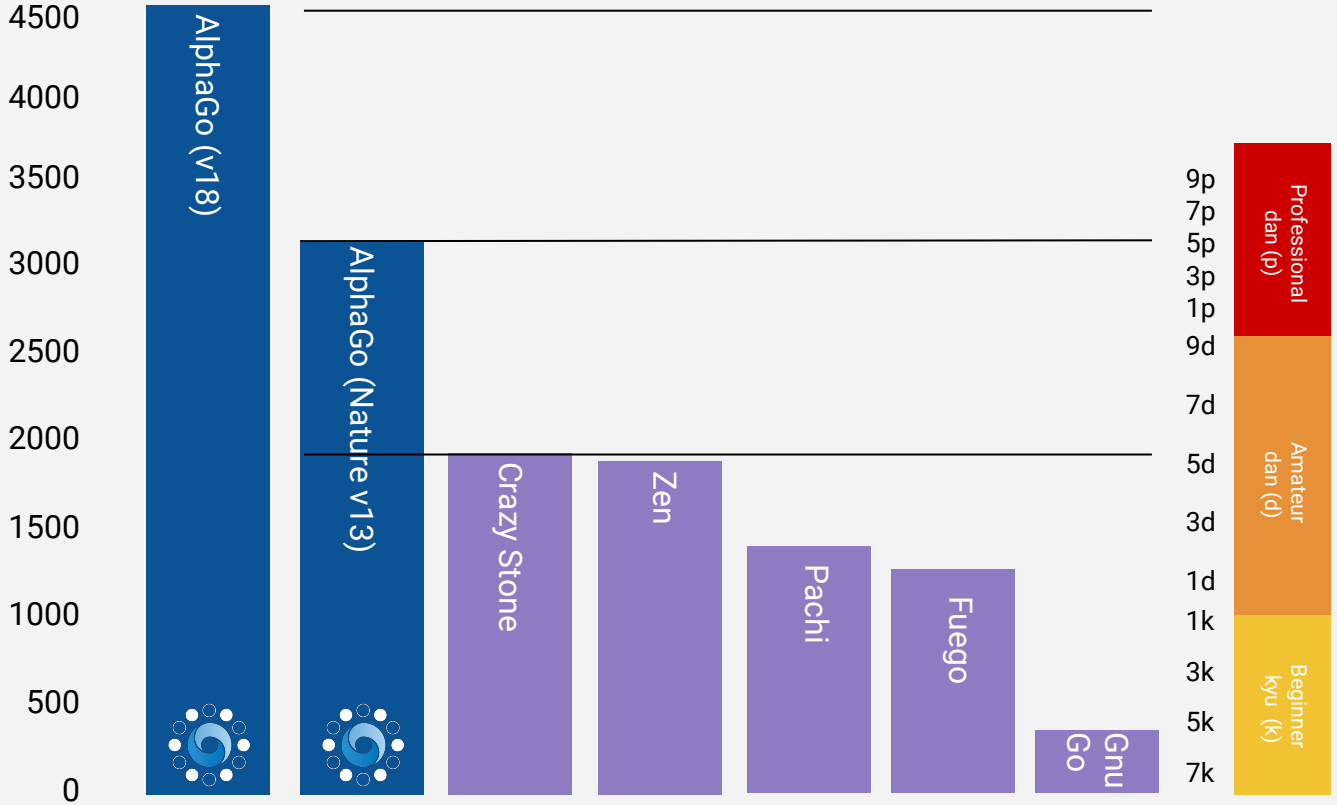


Evaluating current AlphaGo against computers

V13 scores 494/495 against computer opponents

V18 beats V13 3 to 4 stones handicap

CAUTION: ratings based on self-play results



Computer Programs	Calibration	Human Players
AlphaGo (Mar 2016)	*DeepMind challenge match* *4-1*	 Lee Sedol (9p) Top player of past decade
Beats		*Beats*
AlphaGo (Oct 2015)	*Nature match* *5-0*	 Fan Hui (2p) 3-times reigning Euro Champion
Beats		*Beats*
Crazy Stone and Zen	*KGS*	Amateur humans

Extra revision material (Supervised Learning)

- Review of concepts from supervised learning
  - Generalisation, overfitting, Underfitting
  - Learning curves
  - Stochastic gradient descent
- Linear regression
  - Cost function
  - Gradients
- Logistic regression
  - Cost function
  - Gradients

Supervised Learning Problem

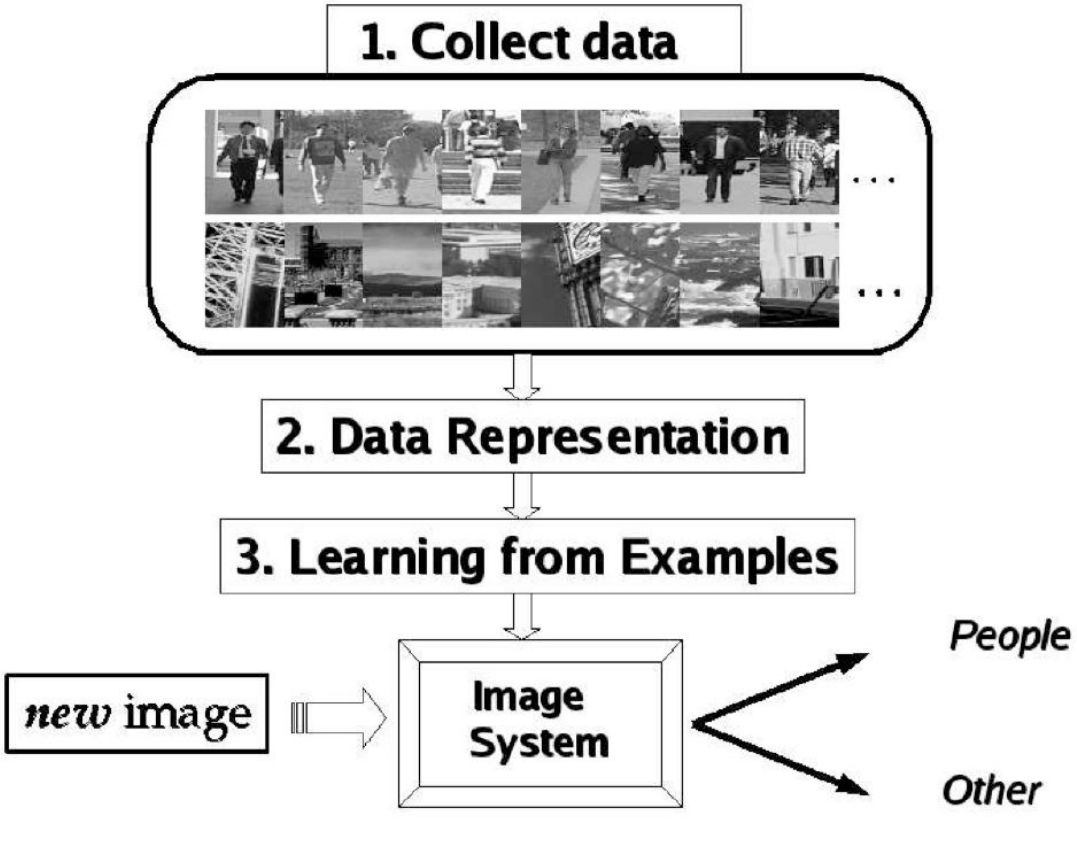
Given a set of **input/output** pairs (**training set**) we wish to compute the functional relationship between the input and the output

Example 1: (people detection) given an image we wish to say if it depicts a person or not. The output is one of two possible categories

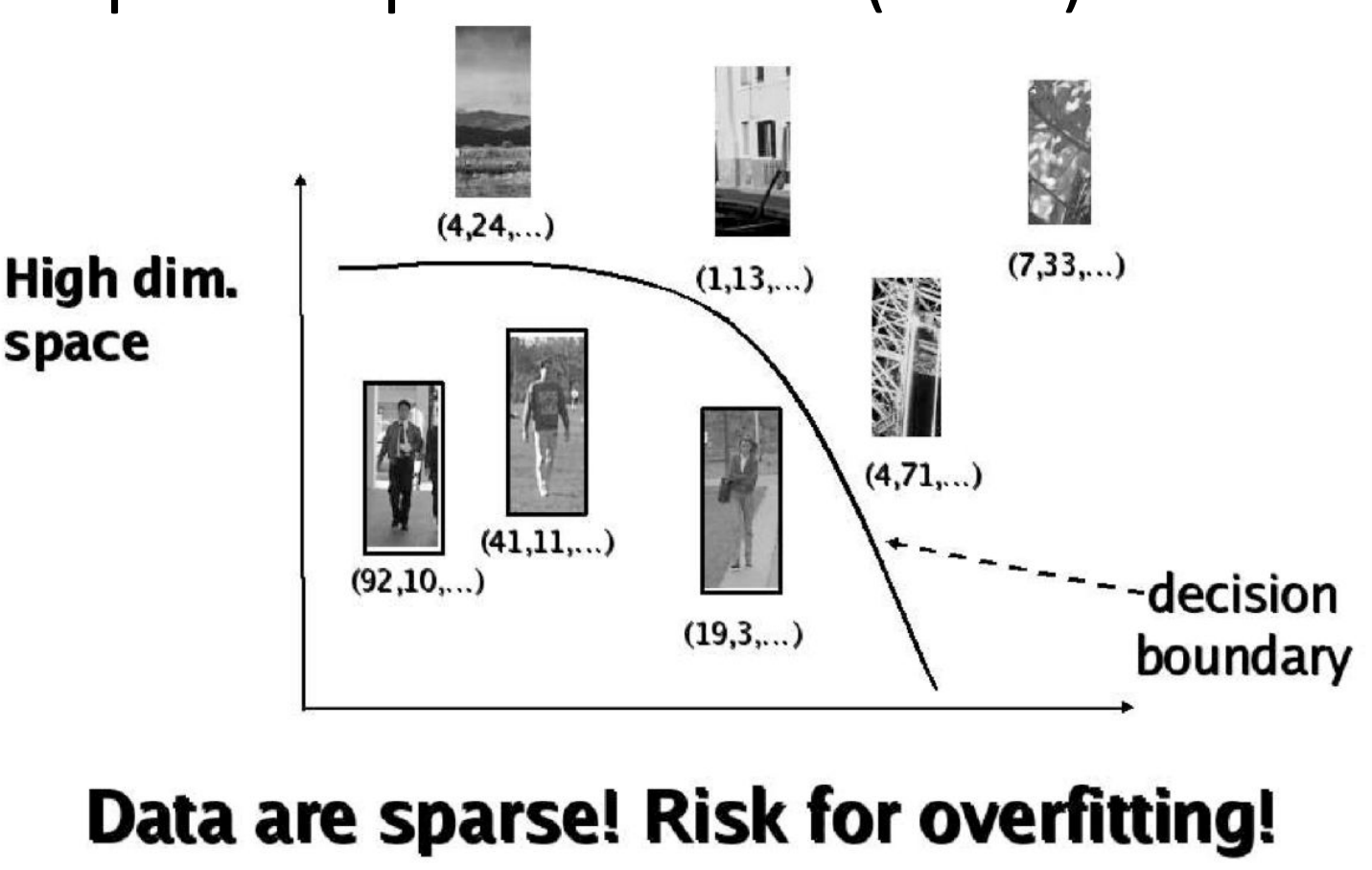
Example 2: (pose estimation) we wish to predict the pose of a face image The output is a continuous number (here a real number describing the face rotation angle)

In both problems the input is a high dimensional vector **x** representing pixel intensity/colour

Example: People Detection



Example: People Detection (cont.)



Supervised Learning Model

- Goal: Given training data (pattern,target) pairs

$$S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$$

infer a function f_S such that

$$f_S(\boldsymbol{x}_i) \approx y_i$$

for the future data

$$S' = \{(\boldsymbol{x}_{m+1}, y_{m+1}), (\boldsymbol{x}_{m+2}, y_{m+2}), \ldots\}.$$

- Classification : $y \in \{-1, +1\}$; Regression : $y \in \mathbb{R}$
- $\mathcal{X}$: input space (eg, $\mathcal{X} \subseteq \mathbb{R}^d$), with elements $\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}_i, \ldots$
- $\mathcal{Y}$: output space, with elements $y, y', y_i, \ldots$

Supervised Learning Problem: Compute a function which best describes I/O relationship

Learning Algorithm

- Training set: $S = \{(\boldsymbol{x}_i, y_i)_{i=1}^m\} \subseteq \mathcal{X} \times \mathcal{Y}$
- A **learning algorithm** is a mapping $S \mapsto f_S$
- A new input $\boldsymbol{x}$ is predicted as $f_S(\boldsymbol{x})$

- Example Algorithms:
  - Linear Regression
  - Logistic Regression
  - Neural Networks
  - Decision Trees
- In this lecture, we will revise linear and logistic regression

Key Questions for the ML Practitioner

- How is the data **collected**? (need assumptions!)
- How do we **represent** the inputs? (may require pre-processing step)
- How **accurate** is the learned function on new data (study of **generalization error**)?
- Many algorithms may exist for a task. How do we choose?
- How "**complex**" is a learning task? (computational complexity, sample complexity)

Important Challenges for ML

- New inputs **differ** from the ones in the training set (look up tables do not work!)
- Inputs are measured with **noise**
- Output is **not deterministically** obtained by the input
- Input is often **high dimensional** but some components/variables may be irrelevant
- How can we incorporate **prior knowledge**?

Generalisation

Most important idea of machine learning:
Train models such that they correctly predict on unseen data
(from the same distribution)

- Empirical risk minimization: Minimise error on training sample
- Validation: Hold out data for testing to obtain unbiased estimator

$$\underbrace{\text{Validation Error}}_{\text{What we care about}} = \underbrace{\text{Training Error}}_{\text{What we optimise}} + \underbrace{(\text{Validation Error} - \text{Training Error})}_{\text{GeneralisationError}}$$

- When data is scarce, can use cross-validation

Cross Validation

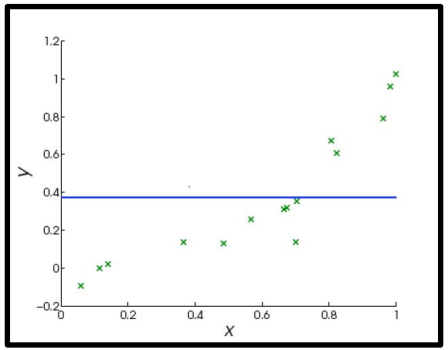
1. we split the data in K parts (of roughly equal sizes)
2. repeatedly train on $K - 1$ parts and test on the part "left out"
3. average the errors of K "validation" sets to give so-called cross-validation error
4. smaller K is less expensive but poorer estimate as size of training set is smaller and random fluctuations larger

For a dataset of size m, m -fold cross-validation is referred to as leave-one-out (LOO) testing

Underfitting and Overfitting

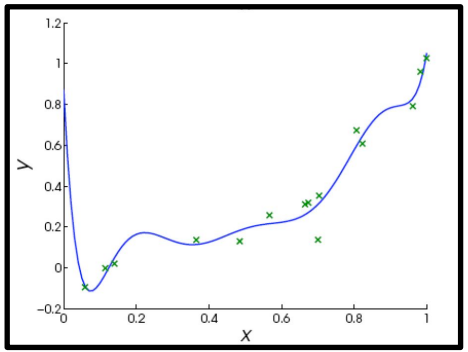
Underfitting

- Error driven by approximation
- High bias / low variance
- What to do?
  - Use more features
  - User more complex model
  - Reduce regularization
  - Train for longer



Overfitting

- Error driven by generalization
- Low bias / high variance
- What to do?
  - Use fewer features
  - Use simpler model
  - Increase regularization
  - Stop training early



More Data versus Better Algorithm

- In high-variance, overfitting situations more data helps
- Example: Confusion Set Disambiguation
- Banko and Brill 2001, “Scaling to Very Very Large Corpora for Natural Language Disambiguation”
- See also: “The Unreasonable Effectiveness of Data”, Pereira, Norvig, Halevy

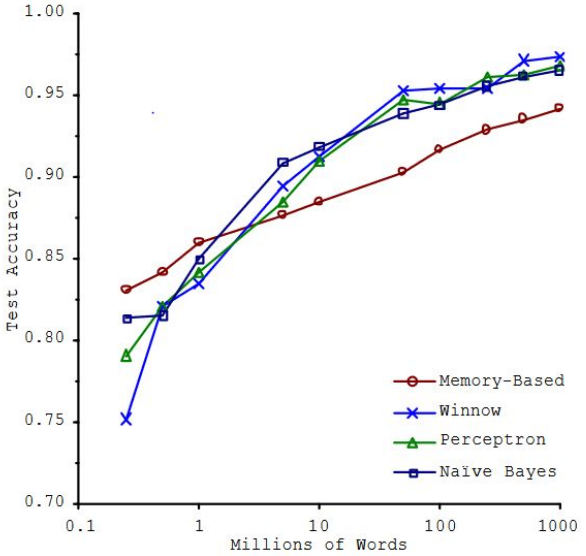
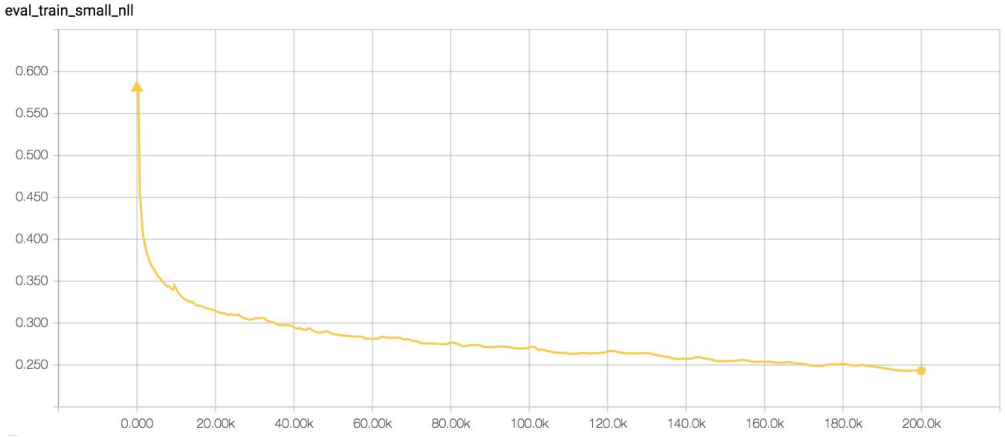


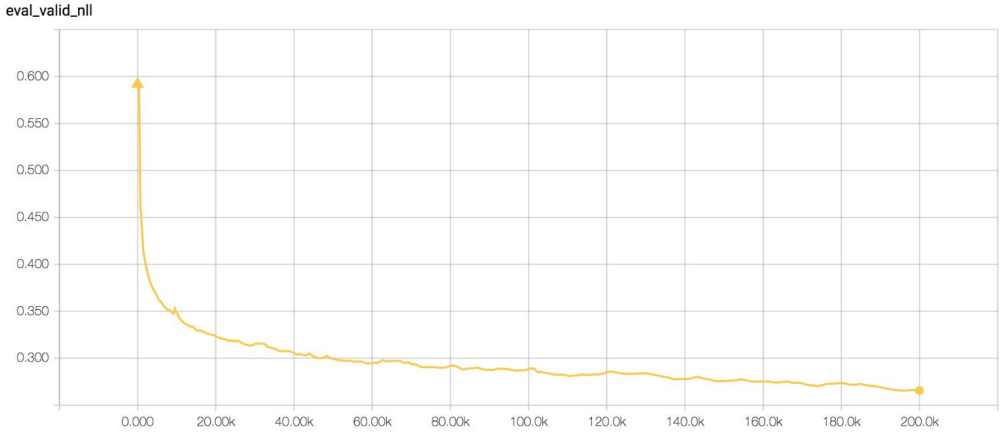
Figure 1. Learning Curves for Confusion Set Disambiguation

Real-World Learning Curves: Underfitting

Training Error



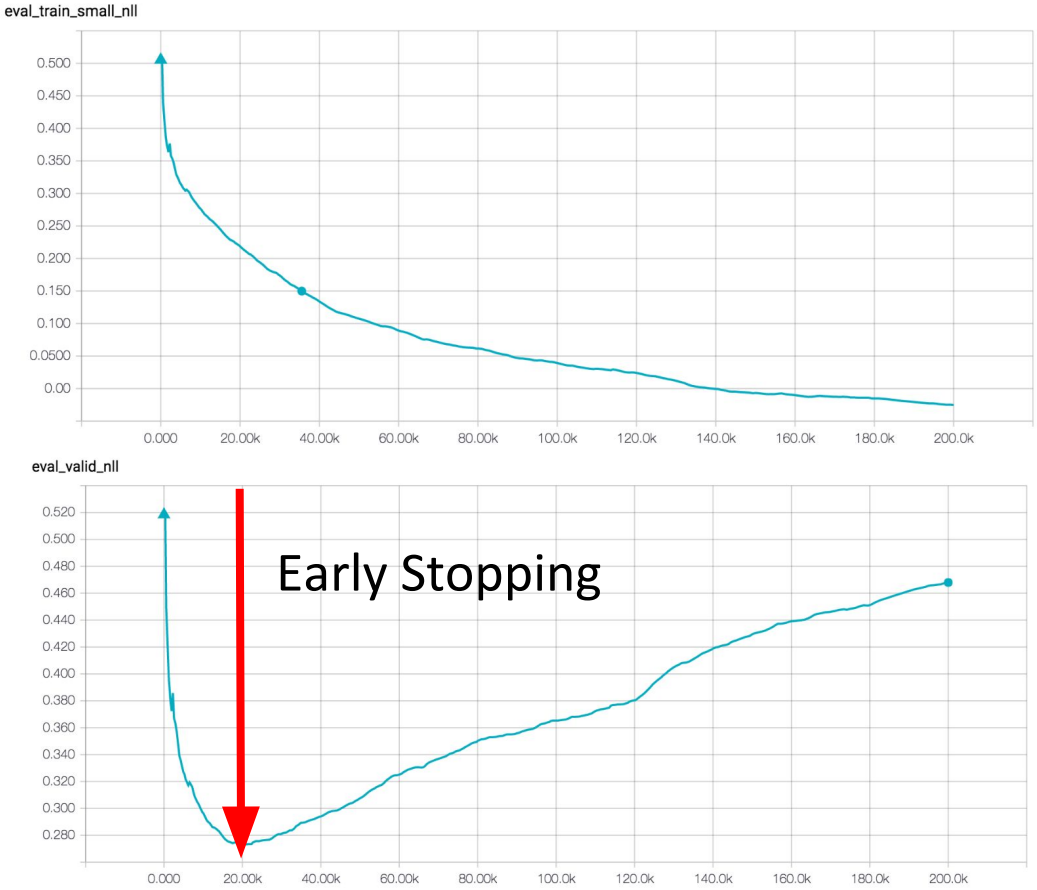
Validation Error



Real-World Learning Curves: Overfitting

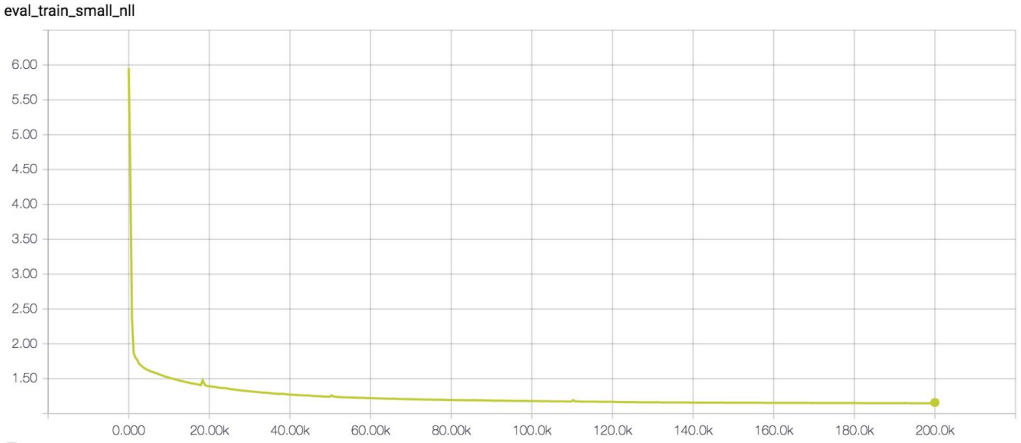
Training Error

Validation Error

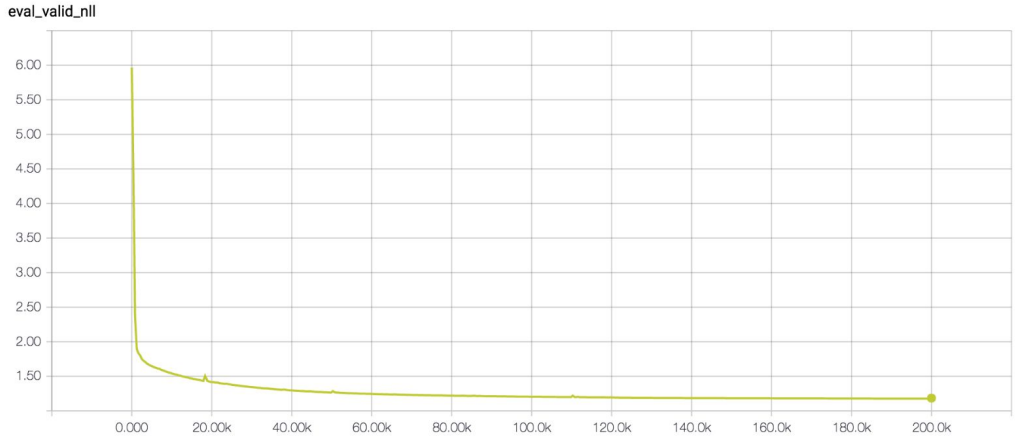


Real-World Learning Curves: Just Right

Training Error

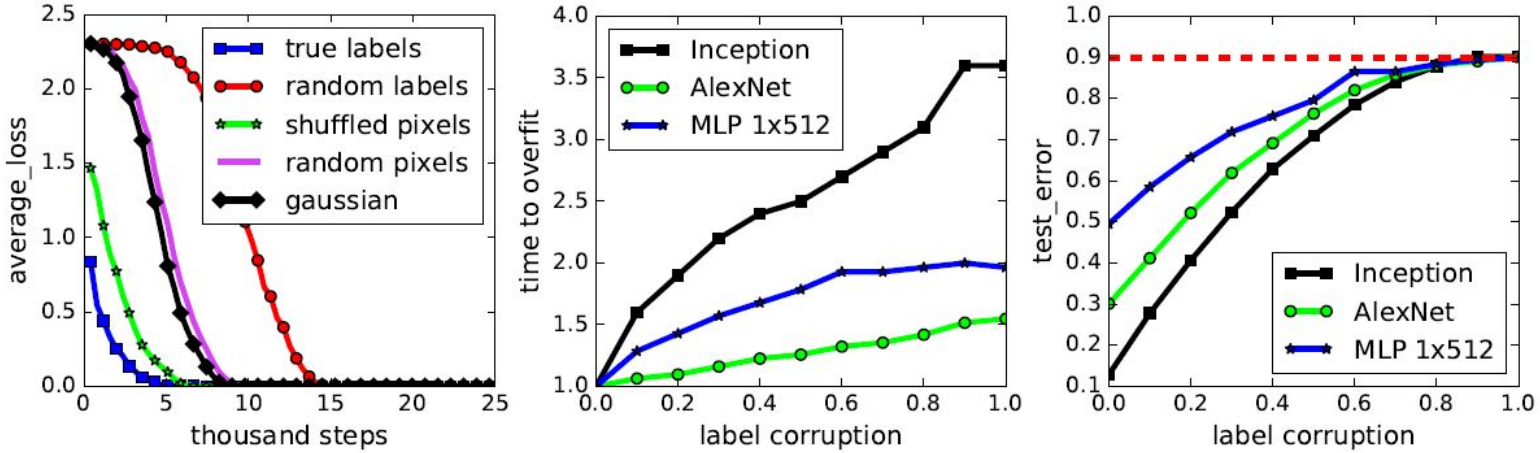


Validation Error



Generalisation in Deep Learning

- "Understanding Deep Learning requires rethinking generalization", Zhang, S. Bengio, Hardt, Recht, Vinyals
- Deep Neural Networks easily fit random labels
- Generalization error varies from 0 to 90% without changes in model
- Deep NNs can even (rote) learn to classify random images



(a) learning curves (b) convergence slowdown (c) generalization error growth

(Stochastic) Gradient Descent

- For linear regression, we can find closed form solution using the (pseudo) matrix inverse (computationally expensive).
- With large data sets or more complex models, this may be impossible
- Batch gradient descent for loss $L(\mathbf{w})$ with learning rate η:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla L(\mathbf{w})$$

- Often better to use stochastic gradient descent for cost functions of the form $L(\boldsymbol{w}) = \sum_{i=1}^{m} f_i(\boldsymbol{w})$:
- For mini-batch S_j apply weight update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \sum_{i \in S_j} \nabla f_i(\mathbf{w})$
- Mini-batch size trades off computational cost and variance.
- Alternative optimization algorithms and analysis in James Martens' lecture

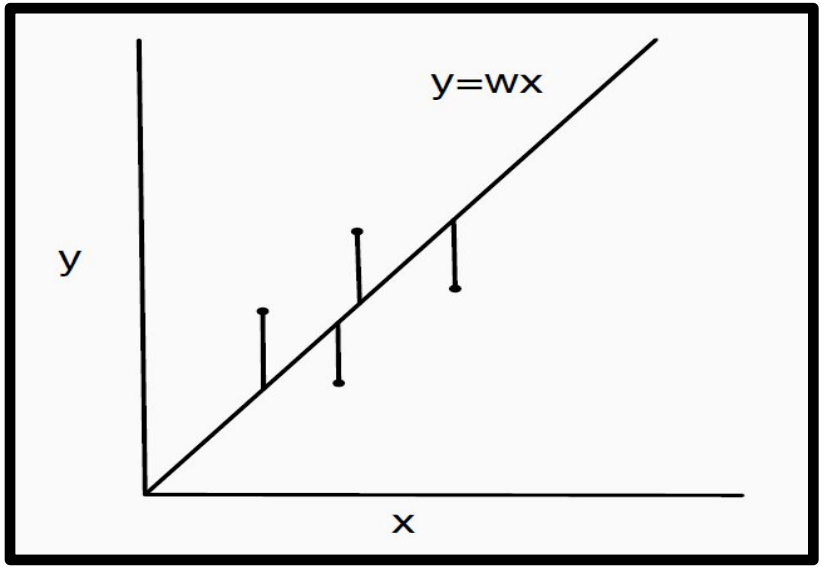
Generalisation from Stochastic Gradient Descent

- “Train faster, generalize better: Stability of Stochastic Gradient Descent”, Moritz Hardt, Benjamin Recht, Yoram Singer

Any model trained with stochastic gradient method in a reasonable amount of time attains small generalization error.

- A learning algorithm is called *stable* if it produces very similar results on two training samples S and S' differing only in one example
- If a learning algorithm is stable its learned model will exhibit good generalization.
- Under certain conditions stochastic gradient descent is stable, hence, the resulting model exhibits good generalisation
- The fewer steps stochastic gradient descent requires, the better the generalization of the resulting model!

Linear Regression



Find a linear predictor $\hat{y} = \boldsymbol{w} \cdot \boldsymbol{x}$ to minimize the square error over the data $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ thus

$$\text{Minimize: } \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)^2$$

Linear Regression Cost Function

- Model: $\hat{y}(\mathbf{w}, \mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
- Example-wise loss function: $l(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$
- Total loss function:

$$L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^m) = \sum_{i=1}^{m} \frac{1}{2}(y_i - \hat{y}(\mathbf{w}, \mathbf{x}_i))^2$$

$$= \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

- Minimising the squared error is equivalent to assuming Gaussian noise in a maximum likelihood estimation

Stochastic gradient descent for regression

- Total loss gradient: $\nabla_{\mathbf{w}} L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^m) = \sum_{i=1}^m \frac{\partial l(y_i, \hat{y})}{\partial \hat{y}} \times \nabla_{\mathbf{w}} \hat{y}(\mathbf{w}, \mathbf{x}_i)$
- Loss gradient: $\frac{\partial l(y, \hat{y})}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \frac{1}{2}(y - \hat{y})^2 = y - \hat{y}$
- Model gradient: $\nabla_{\mathbf{w}} \hat{y}(\mathbf{w}, \mathbf{x}) = \nabla_{\mathbf{w}} (\mathbf{w} \cdot \mathbf{x}) = \mathbf{x}$
- Put together:

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^m) &= \sum_{i=1}^m (y_i - \hat{y}(\mathbf{w}, \mathbf{x})) \times \mathbf{x}_i \\ &= \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i) \times \mathbf{x}_i \end{aligned}$$

Batch and stochastic gradient descent

- Batch gradient descent (entire batch of data):

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^{m}) \quad = \quad \mathbf{w} - \eta \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i) \times \mathbf{x}_i$$

- Online gradient descent (one by one) :

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} l(\mathbf{w}, \mathbf{x}_j, y_j) \quad = \quad \mathbf{w} - \eta (y_j - \mathbf{w} \cdot \mathbf{x}_j) \times \mathbf{x}_j$$

- Often best to use mini-batch ($1 < k < m$ examples at a time)

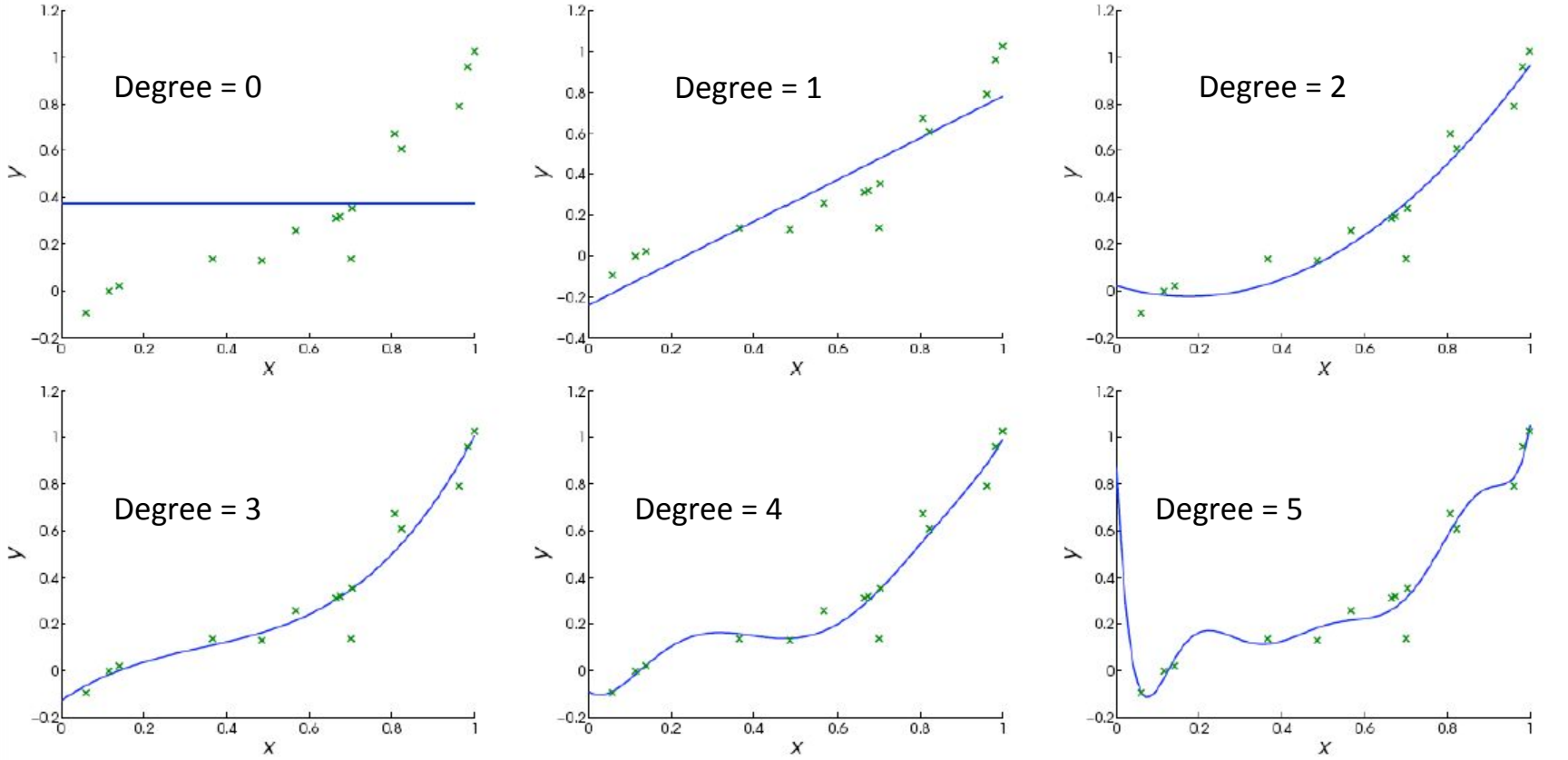
Regularisation

- One way of limiting the capacity of models is regularization
- The most popular regulariser for linear models and neural networks is called weight decay
- In the cost function, one adds a term $\lambda||\boldsymbol{w}||^2$ to penalize large weights
- In the gradient update rule, this leads to a “force” that aims to reduce the length of the weight vector
- Alternatively, in a Bayesian maximum aposteriori view, the same effect can be achieved with a zero mean Gaussian prior over weights
- Other regularisers can induce sparse structure in the weights (L1) or reduce co-dependence (dropout) → more in Simon’s, Karen’s lectures

Non-linear Basis Functions

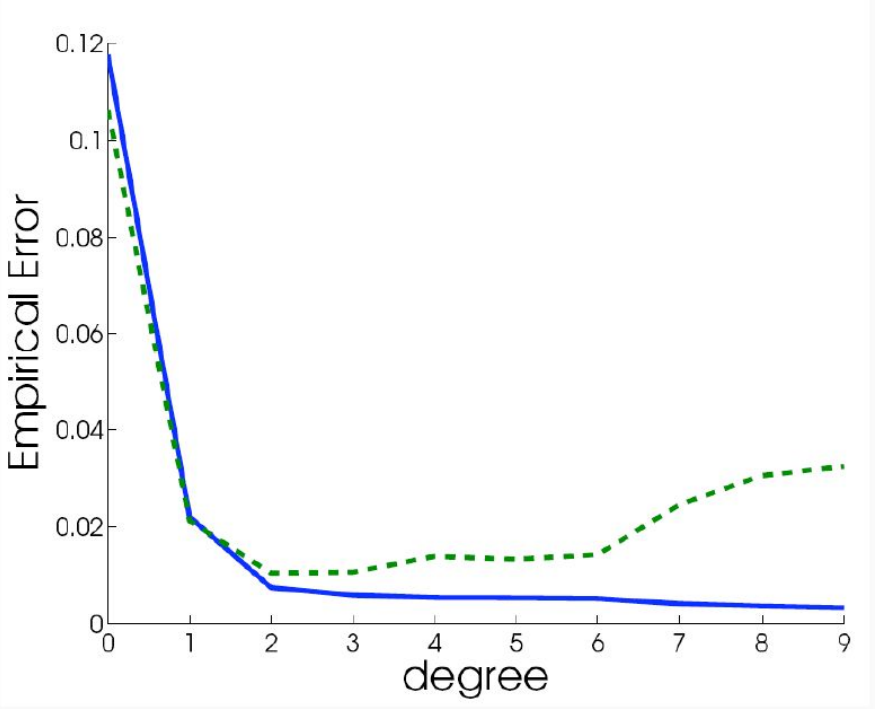
- Linear regression is linear in the parameters, not necessarily in the inputs!
- Can use basis functions $\phi(x)$ to map data into a different feature space representation.
- Example: $\phi_0(x) = 1$, $\phi_1(x) = (1, x)$, $\phi_2(x) = (1, x, x^2)$, etc.
- Now linear model can be written as $\hat{y} = \boldsymbol{w} \cdot \phi(x)$ and is non-linear in the data
- This provides a richer hypothesis space for fitting the data
- With kernels $k(x, x')$ we can avoid explicit mapping (SVMs)

Regression with polynomial basis functions



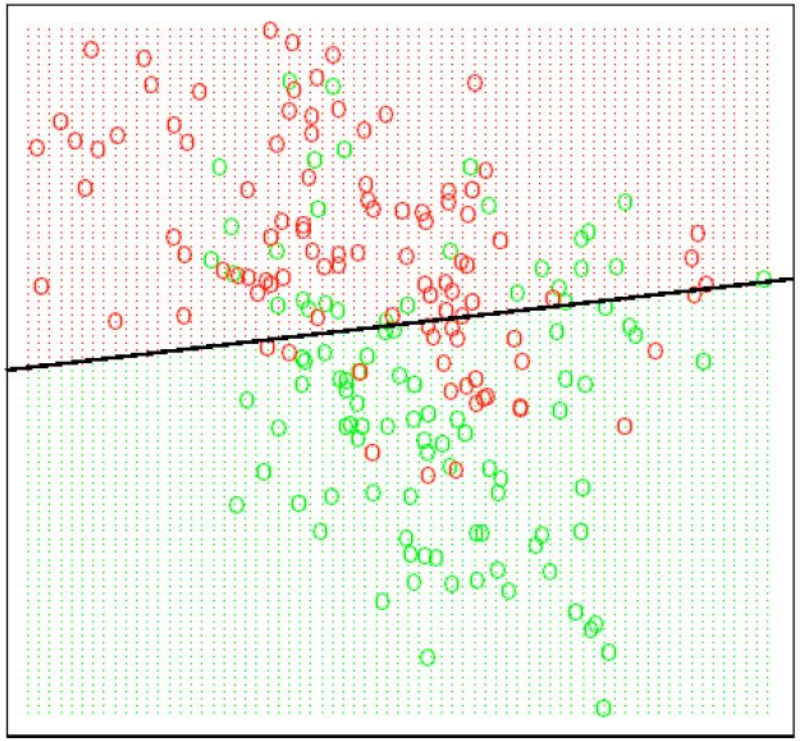
Polynomial Fit for different degrees

- Training error goes down with increasing degree (better fit)
- Test error is optimal at degree 2, and deteriorates for higher degrees
- Note the similarity to learning curves discussed earlier. The effective hypothesis class of neural networks becomes more complex with longer training



Logistic Regression for classification

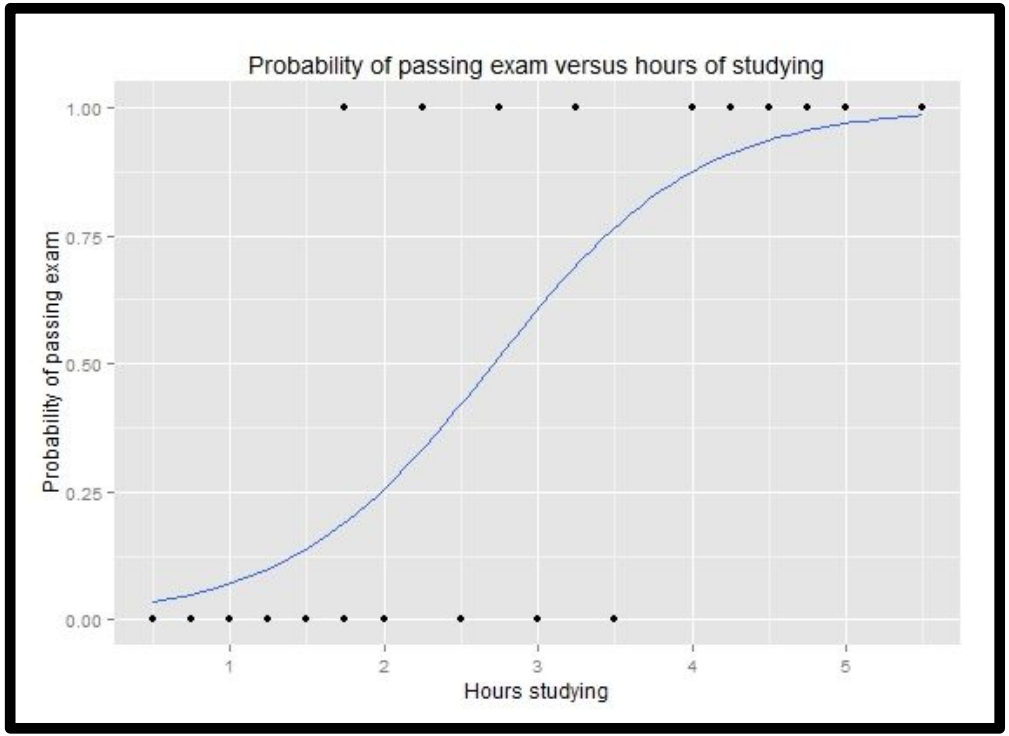
- Generalized linear model for binary classification
- Used, e.g., in click-through-rate prediction for search engine advertising
- Find linear hyperplane to separate the data
- Predict probability of class



Logistic Regression Cost Function

- Linear model: $z(\mathbf{w}, \mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
- (Inverse) Link function: $\hat{p}(z) = \frac{1}{1 + \exp(-z)}$
- Cross entropy loss: $l(y, \hat{p}) = y \log \hat{p} + (1 - y) \log(1 - \hat{p})$
- The regression loss is a composition of these three functions, aggregated over training examples

Logistic (Inverse) Link Function



Cross Entropy

- The general form of cross entropy between distributions p and q is given by $H(p,q) = E_p[-\log q] = H(p) + D_{KL}(p||q)$
- Cross entropy measures how many bits are needed to encode a message assuming that the letters are drawn from distribution q when the real distribution is p.
- In the case of two point discrete distributions with success probabilities p and $1-p$, cross entropy is given by

$$H(p,q) = -p\log q \ -(1-p)\log(1-q)$$

Logistic Regression Cost Function

$$
\begin{aligned}
L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^m) &= -\sum_{i=1}^{m} y_i \log \hat{p}(z_i) + (1 - y_i) \log(1 - \hat{p}(z_i)) \\
&= -\sum_{i=1}^{m} y_i \log \left(\frac{1}{1 + \exp(-z_i)} \right) + (1 - y_i) \log \left(1 - \frac{1}{1 + \exp(-z_i)} \right) \\
&= -\sum_{i=1}^{m} y_i \log \left(\frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)} \right) + (1 - y_i) \log \left(1 - \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)} \right) \\
&= \sum_{i=1}^{m} \log(1 + \exp(\mathbf{w} \cdot \mathbf{x}_i)) - y_i \mathbf{w} \cdot \mathbf{x}_i
\end{aligned}
$$

Modular Gradients for Logistic Regression

- Total Gradient:

$$\nabla_{\mathbf{w}} L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^m) = \sum_{i=1}^m \left.\frac{\partial l(y_i, \hat{p})}{\partial \hat{p}}\right|_{\hat{p}=\hat{p}(z_i)} \times \left.\frac{\partial \hat{p}(z)}{\partial z}\right|_{z=z_i} \times \nabla_{\mathbf{w}} z(\mathbf{w}, \mathbf{x})|_{\mathbf{x}=\mathbf{x}_i}$$

- Loss gradient:

$$\left.\frac{\partial l(y_i, \hat{p})}{\partial \hat{p}}\right|_{\hat{p}=\hat{p}(z_i)} = -\frac{y_i}{\hat{p}(z_i)} + \frac{1-y_i}{1-\hat{p}(z_i)}$$

- Link gradient:

$$\left.\frac{\partial \hat{p}(z)}{\partial z}\right|_{z=z_i} = \left.\frac{\exp(-z)}{(1+\exp(-z))^2}\right|_{z=z_i} = \hat{p}(z_i)(1-\hat{p}(z_i))$$

- Model gradient:

$$\nabla_{\mathbf{w}} z(\mathbf{w}, \mathbf{x})|_{\mathbf{x}=\mathbf{x}_i} = \nabla_{\mathbf{w}} \mathbf{w} \cdot \mathbf{x}|_{\mathbf{x}=\mathbf{x}_i} = \mathbf{x}_i$$

Putting the gradient back together

$$
\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^m) &= \sum_{i=1}^{m} \left(-\frac{y_i}{\hat{p}(z_i)} + \frac{1-y_i}{1-\hat{p}(z_i)} \right) \times \hat{p}(z_i)(1-\hat{p}(z_i)) \times \mathbf{x}_i \\
&= (\hat{p}(z_i) - y_i)\mathbf{x}_i \\
&= \left(\frac{1}{1+\exp(-z_i)} - y_i \right) \mathbf{x}_i \\
&= \left(\frac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x}_i)} - y_i \right) \mathbf{x}_i
\end{aligned}
$$

- Similarly, the backpropagation algorithm works through the layers of deeper neural networks to calculate error gradients w.r.t. to weights
- Simon's lecture will give more details